Continual Curiosity-Driven Skill Acquisition from High-Dimensional Video Inputs for Humanoid Robots

Varun Raj Kompella, Marijn Stollenga, Matthew Luciw, Juergen Schmidhuber

The Swiss AI Lab IDSIA, USI & SUPSI, Galleria 2, 6928 Manno-Lugano, Switzerland

Abstract

In the absence of external guidance, how can a robot learn to map the many raw pixels of high-dimensional visual inputs to useful action sequences? We propose here Continual Curiosity driven Skill Acquisition (CCSA). CCSA makes robots intrinsically motivated to acquire, store and reuse skills. Previous curiosity-based agents acquired skills by associating intrinsic rewards with world model improvements, and used reinforcement learning to learn how to get these intrinsic rewards. CCSA also does this, but unlike previous implementations, the world model is a set of compact low-dimensional representations of the streams of high-dimensional visual information, which are learned through incremental slow feature analysis. These representations augment the robot's state space with new information about the environment. We show how this information can have a higher-level (compared to pixels) and useful interpretation, for example, if the robot has grasped a cup in its field of view or not. After learning a representation, large intrinsic rewards are given to the robot for performing actions that greatly change the feature output, which has the tendency otherwise to change slowly in time. We show empirically what these actions are (e.g., grasping the cup) and how they can be useful as skills. An acquired skill includes both the learned actions and the learned slow feature representation. Skills are stored and reused to generate new observations, enabling continual acquisition of complex skills. We present results of experiments with an iCub humanoid robot that uses CCSA to incrementally acquire skills to topple, grasp and pick-place a cup, driven by its intrinsic motivation from raw pixel vision.

Keywords: Reinforcement Learning, Artificial Curiosity, Skill Acquisition, Slow Feature Analysis, Continual Learning, Incremental Learning, iCub

1. Introduction

Over the past decade, there has been a growing trend in humanoid robotics research towards robots with a large number of joints, or degrees of freedom, notably the ASIMO [1], PETMAN [2] and the iCub [3]. These robots demonstrate a high

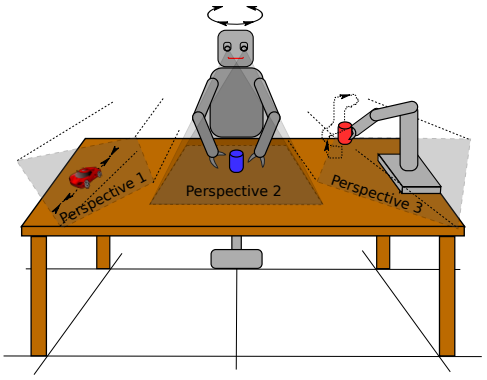

Figure 1: A playroom scenario for a baby humanoid-robot in a lab environment, where it is placed next to a table with a few moving objects. The robot has a limited field-of-view and encounters continuous streams of images as it holds or shifts its gaze. Figure shows three such perspectives oriented towards the moving objects. How can the robot learn to solve tasks in the absence of an external guidance?

amount of dexterity and are potentially capable of carrying out complex human-like manipulation. When interacting with the real-world, these robots are faced with several challenges, not the least of which is the problem of how to solve tasks upon processing an abundance of high-dimensional sensory data.

In the case of well structured environments, these robots can be carefully programmed by experts to solve a particular task. But real-world environments are usually unstructured and dynamic, which makes it is a daunting task to program these robots manually. This problem can be substantially alleviated by using reinforcement learning (RL; [4, 5]), where a robot learns to acquire desired task-specific behaviors, by maximizing the accumulation of task-dependent external rewards through simple trial-and-error interactions with the environment.

Unfortunately, for humanoid robots equipped with vision, the sensory and joint state space is so large that it is extremely difficult to procure the rewards (if any exist) by random exploration. For example, if the robot receives a reward for sorting objects, it could take an extremely long time to obtain the reward for the first time. Therefore, it becomes necessary to (a) build lower-dimensional representations of the state-space to make learning tractable and (b) to explore the environment efficiently. But how can these robots learn to do this in the presence of external rewards that are typically only sparsely available?

Much of the human capacity to explore and solve problems is driven by self-supervised learning [6, 7], where we seek to acquire behaviors by creating novel situations and learning from them. As an example, consider a simple playroom scenario for a baby humanoid as shown in Figure 1. Here, the robot is placed

next to a table with a few moving objects. The robot has a limited field-of-view and encounters continuous streams of images as it holds or shifts its gaze. If the robot can learn compact representations and predictable behaviors (*e.g.,* to grasp) from its interactions with the cup, then by using these learned behaviors, it can speed up the acquisition of external rewards related to some teacher-defined task, such as placing the cup at a particular location. Continually acquiring and reusing a repertoire of behaviors and representations of the world, learned through self-supervision, can therefore make the robot adept in solving many external tasks.

But how can the robot (a) self-supervise its exploration, (b) build representations of the high-dimensional sensory inputs and (c) continually acquire skills that enable it to solve new tasks? These problems have individually been researched in the machine learning and robotics literature [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29]. However, to develop a *single* system that addresses all these important issues together is a challenging open problem in artificial intelligence (AI) research. We propose an online-learning framework that addresses this open problem.

In order to make the robot self-supervised or intrinsically-motivated to explore new environments, we use the theory of Artificial Curiosity (AC; [30, 31]). AC mathematically describes curiosity and creativity. AC-driven agents are interested in the learnable but as-yet-unknown aspects of their environment, and are disinterested in the already learned and inherently unlearnable (noisy) aspects. Specifically, the agent receives *intrinsic rewards* for action sequences, and these rewards are proportional to the improvement of the agent's internal model or predictor of the environment. Using RL and the self-generated intrinsic rewards derived using AC [32, 33, 34, 35, 36, 25], the agent is motivated to explore the environment where it makes maximum learning progress.

Most RL algorithms however, tend to work only if the dimensionality of the state space is small, or its structure is simple. In order to deal with massive high-dimensional streams of raw sensory information obtained, for example through vision, it is essential to reduce the input dimensionality by building low-dimensional but informative *abstractions* of the environment [37]. An *abstraction* maps the high-dimensional input to a low-dimensional output. The high-dimensional data sensed by a robot is often temporally correlated and can be greatly compressed if the temporal coherence in the data is exploited. Slow Feature Analysis (SFA; [14, 38, 39]) is an unsupervised learning algorithm that extracts temporal regularities from rapidly changing raw sensory inputs. SFA is based on the Slowness Principle [40, 41, 42], which states that the underlying causes of changing signals vary more slowly than the primary sensory stimulus. For example, individual retinal receptor responses or gray-scale pixel values of video may change quickly compared to latent abstract variables, such as the position of a moving object. SFA has

achieved success in many problems and scenarios, e.g., extraction of driving forces of a dynamical system [43], nonlinear blind source separation [44], as a preprocessor for reinforcement learning [39], and learning of place-cells, head-direction cells, grid-cells, and spatial view cells from high-dimensional visual input [38].

SFA techniques are not readily applicable to open-ended online learning agents, as they estimate covariance matrices from the data via batch processing. We instead use Incremental Slow Feature Analysis (IncSFA; [45, 46]), which does not need to store any input data or computationally expensive covariance matrix estimates. IncSFA makes it feasible to handle high-dimensional image data in an open-ended manner.

IncSFA, like most online learning approaches, gradually forgets previously learned representations whenever the statistics of the input change, for example, when the robot shifts its gaze from perspective two to perspective one in Figure 1. To address this issue, in our previous work, we proposed an algorithm called Curiosity-Driven Modular Incremental Slow Feature Analysis (Curious Dr. MISFA; [47, 48]), which retains what was previously learned in the form of *expert modules* [29]. From a set of input video streams, Curious Dr. MISFA actively learns multiple expert modules comprising slow feature abstractions, in the order of increasing learning difficulty. The algorithm continually estimates the initially unknown learning difficulty through intrinsic rewards generated by exploring the input streams.

Using Curious Dr. MISFA, the robot in Figure 1 finds its interactions with the plastic cup more interesting (easier to encode) than the complex movements of the other objects. This results in a compact slow feature abstraction that encodes its interactions with the cup. Eventually, the robot finds the cup-interaction boring and its interest shifts towards encoding other perspectives while retaining the learned abstraction. Can the robot simultaneously acquire re-usable skills while acquiring abstractions? Each abstraction learned encodes some previously unknown regularity in the input observations, which can therefore be used as a basis for acquiring new skills.

Our contribution here is the Continual Curiosity-driven Skill Acquisition (CCSA) framework, for acquiring both abstractions and skills in an online and continual manner. In RL, the *options* framework [49] formalizes skills as RL policies, active within a subset of the state space, which can terminate at subgoals, after which another option takes over. When the agent has a high-dimensional input, like vision, an option requires a dimensionality reducing abstraction, so that policy learning becomes tractable. CCSA is a task-independent curiosity-driven learning algorithm that combines Curious Dr. MISFA with the options framework. Each slow feature abstraction learned by Curious Dr. MISFA augments the robot's default state space, which in our case is a set of low-level kinematic joint poses learned using Task Rel-

evant Roadmaps [50]. This augmented state space is then clustered to create new distinct states. A Markovian transition model is learned by exploring the new state space. The reward function is also learned through exploration, with the agent being intrinsically rewarded for making state-transitions that produce a large variation in the slow-feature outputs. This specialized reward function is used to build the option's policies, to drive the robot to states where such transitions will occur. Such transitions are shown to correspond to *bottleneck states*, i.e., "doorways", which are known to be good subgoals in the absence of externally imposed goals [51, 52]. Once the transition and reward functions are learned, the option's policy is learned via Least-Squares Policy Iteration [53]. Skills acquired by the robot in the form of options, are reused to generate new input observations, enabling acquisition of more complex skills in a continual open-ended manner [29, 54]. Using CCSA, in our experiments, an iCub humanoid robot addresses the open problems discussed earlier, acquiring a repertoire of skills (topple, grasp) from raw-pixel vision, driven purely by its intrinsic motivation.

The rest of this paper is organized as follows. Section 2 discusses related research work carried out prior to this paper. Sections 3 and 4 present an overview and a formulation of the learning problem associated with the CCSA framework. Section 5 discusses details of the internal workings of CCSA. Section 6 contains experiments and results conducted using an iCub humanoid robot; Sections 7-8 presents future work and conclusions.

2. Related Work

Existing intrinsically-motivated skill acquisition techniques in RL have been applied to simple domains. For example, Bakker and Schmidhuber [55] proposed a hierarchical RL framework called HASSLE in a grid world environment, where high-level policies discover subgoals from clustering distance-sensor outputs and low-level policies specialize on reaching the subgoals. Stout and Barto [24] explore the use of a competence-based intrinsic motivation as a developmental model for skill acquisition in simple artificial grid-world domains. Pape et al. [25] proposed a method for autonomous acquisition of tactile skills on a biomimetic robot finger, through curiosity-driven reinforcement learning.

There have been attempts to find skills using feature-abstractions in domains such as those of humanoid robotics. Hart [56] proposed an intrinsically motivated hierarchical skill acquisition approach for a humanoid robot. The system combines *discrete event dynamical systems* [57] as a control basis and an intrinsic reward function [26] to learn a set of controllers. However, the intrinsic reward function used is task specific, and the system requires a teacher to design a developmental schedule for the robot.

Konidaris et al. [58, 59] show how each option might be assigned with an abstraction from a library of many sensorimotor abstractions to acquire skills. The abstractions have typically been hand-designed and learning was assisted by human-demonstration. In their recent work [27], an intrinsic motivation system makes a robot acquire skills from one task to improve the performance on a second task. However, the robot used augmented reality tags to identify target objects and had access to a pre-existing abstraction library. CCSA autonomously learns a library of abstractions and control policies simultaneously from raw-pixel streams generated via exploration, without any prior-knowledge of the environment.

Mugan and Kuipers's [60] Qualitative Learner of Action and Perception system discretizes low-level sensorimotor experience through defining landmarks in the variables and observing contingencies between landmarks. It builds predictive models on this low-level experience, which it later uses to generate plans of actions. It either selects its actions randomly (early) or such that it expects to make fast progress in the performance of the predictive models (artificial curiosity). The sensory channels are preprocessed so that the input variables, for example, track the positions of the objects in the scene. A major difference between this system and ours is that we operate upon the raw pixels directly, instead of assuming the existence of a low-level sensory model that can track the positions of the objects in the scene.

Baranes and Oudeyer [61] proposed an intrinsic motivation architecture called SAGG-RIAC, for adaptive goal-exploration. The system comprises two learning parts, one for self-generation of subgoals within the task-space and the other for exploration of low-level actions to reach the subgoals selected. The subgoals are generated using heuristic methods based on a local measure of *competence progress*. The authors show results using a simulated quadruped robot on reaching tasks. The system however, assumes that a low-dimensional task-space is provided. CCSA is a task-independent approach, where subgoals are generated automatically by the slow feature abstractions that encode spatio-temporal regularities in the raw high-dimensional video inputs.

Ngo et al. [62, 63] investigated an autonomous learning system that utilizes a progress-based curiosity drive to ground a given abstract action, e.g., placing an object. The general framework is formulated as a selective sampling problem in which an agent samples any action in its current situation as soon as it sees that the effects of this action are *statistically* unknown. If no available actions have a statistically unknown outcome, the agent generates a plan of actions to reach a new setting where it expects to find such an action. Experiments were conducted using a Katana robot arm with a fixed overhead camera, on a block-manipulation task. The authors show that the proposed method generates sample-efficient curious exploratory behavior and continual skill acquisition. However, unlike CCSA,

the sensorimotor abstractions are hand-designed and not learned by the agent.

CCSA uses IncSFA to find low-dimensional manifolds within the raw pixel inputs, providing a basis for coupled perceptual *and* skill learning. We emphasize the special utility of SFA for this task over similar methods such as principal component analysis [64], or predictive-projections [65], which are based on variance or nearest neighbor learning, while Slow features through IncSFA extract temporal invariance from input streams that represent "doorway" or "bottleneck" aspects (choke-points between two more fully connected subareas), similar to Laplacian-Eigen Maps [66, 67, 68]. The hierarchical reinforcement learning literature [69, 70, 71, 49, 51, 55, 67, 52] illustrates that such bottlenecks can be useful subgoals. Finding such bottlenecks in visual input spaces is a relatively new concept, and one we exploit in the iCub experiments. For example, while it moves its arm around a cup in the scene, the bottleneck state is where it topples the cup over, invariant to the arm position. The two subareas in this case are 1. the cup is upright (stable) while the arm moves around, 2. the cup is on its side (stable) while the arm moves around. More studies on the types of representations learned by the IncSFA algorithm can be found elsewhere [47, 46].

An initial implementation of Curious Dr. MISFA for learning slow feature abstractions [48], a discussion on its neurophysiological correlates [47] and a prototypical construction of a skill from a slow feature abstraction [72] can be found in our previous work. The novel contribution of this paper is that we present an online learning algorithm (CCSA) that uses Curious Dr. MISFA for learning slow feature abstractions, such that it enables a robot to acquire, store and reuse skills in an open-ended *continual* manner. We also formally address the underlying learning problem of task-independent continual curiosity-driven skill acquisition. We demonstrate the working of our algorithm with iCub experiments and show the advantages of intrinsically motivated skill acquisition for solving an external task.

3. Overview of the Proposed Framework

In this section, we will briefly summarize the overall framework of the proposed algorithm, which we call Continual Curiosity driven Skill Acquisition (CCSA). Figure 2 illustrates the overall framework. The learning problem associated with CCSA can be described as follows: From a set of pre-defined or previously acquired input exploratory behaviors, which generate potentially high-dimensional time-varying observation streams, the objective of the agent is to (a) acquire an easily learnable yet unknown target behavior and (b) re-use the target behavior to acquire more complex target behaviors. The target behaviors represent the skills acquired by the agent. A sample run of the CCSA framework to acquire a skill is as follows (see Figure 2):

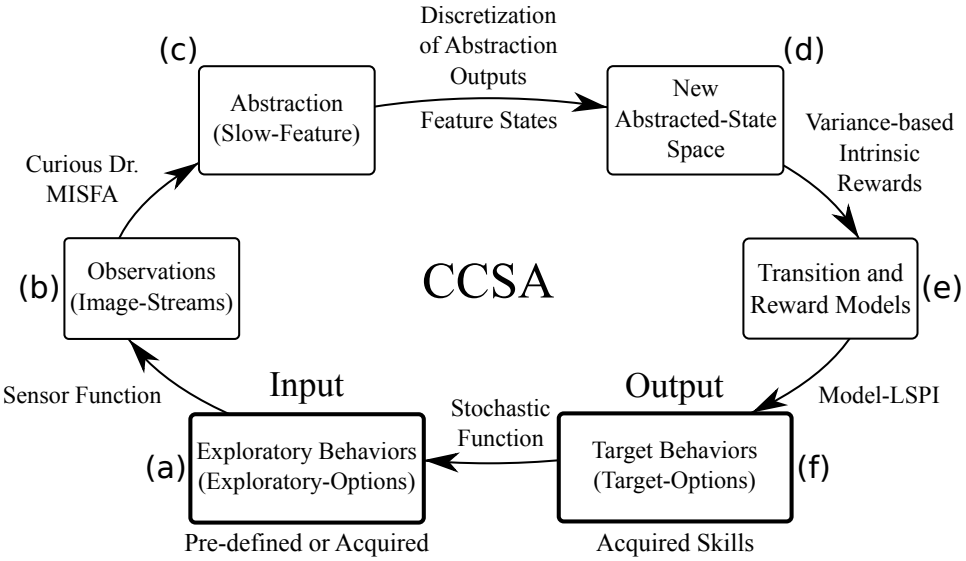

Figure 2: High-level control flow of the Continual Curiosity-driven Skill Acquisition (CCSA) framework. (a) The agent starts with a set of pre-defined or previously acquired exploratory behaviors (represented as exploratory options). (b) It makes high-dimensional observations upon actively executing the exploratory options. (c) Using the Curious Dr. MISFA algorithm, the agent learns a slow feature abstraction that encodes the easiest-to-learn yet unknown regularity in the observation streams. (d) The slow feature abstraction outputs are clustered to create feature states that are augmented to the agent's abstracted-state space. (e) A Markovian transition model of the new abstracted-state space and an intrinsic reward function are learned through exploration. (f) A deterministic policy is then learned via model-based Least-Squares Policy Iteration (Model-LSPI) and a target option is constructed. The deterministic target-option's policy is modified to a stochastic policy in the agent's new abstracted states and is added to the set of exploratory options.

(a) The agent starts with a set of pre-defined or previously acquired exploratory behaviors. We make use of the *options* framework [49] to formally represent the exploratory behaviors as exploratory options (see Section 4 for a formal definition of the terminology used here).

(b) The agent makes high-dimensional observations through a sensor-function, such as a camera, upon actively executing the exploratory options.

(c) Using our previously proposed curiosity-driven modular incremental slow feature analysis (Curious Dr. MISFA) algorithm, the agent learns a slow feature abstraction that encodes the easiest-to-learn yet unknown regularity in the observation streams (see Section 5.2).

(d) The slow feature abstraction outputs are clustered to create feature states that are augmented to the agent's abstracted-state space, which contains previously encoded feature-states (see Section 5.3).

(e) A Markovian transition model is learned by exploring the new abstracted-state space. The reward function is also learned through exploration, with the agent being intrinsically rewarded for making state-transitions that produce a large variation (high statistical variance) in the slow-feature outputs. This specialized reward function is used to learn action-sequences (policy) that drives the agent to states where such transitions will occur (see Section 5.3).

(f) Once the transition and reward functions are learned, a *deterministic* policy is learned via model-based Least-Squares Policy Iteration (LSPI; [53]). The learned policy and the learned slow feature abstraction together constitute a target option, which represents the acquired skill (see Section 5.3).

(f)-(a) The deterministic target-option's policy is modified to a stochastic policy in the agent's new abstracted states and is added to the set of exploratory options (see Section 5.4). This enables the agent to reuse the skills to acquire more complex skills in a continual open-ended manner [29, 54].

CCSA is a task-independent algorithm, *i.e.*, it does not require any design modifications when the environment is changed. However, CCSA makes the following assumptions: (a) The agent's default abstracted-state space contains low-level kinematic joint poses of the robot learned offline using Task Relevant Roadmaps [50]. This is done to limit the iCub's exploration of its arm to a plane parallel to the table. This assumption can be relaxed resulting in a larger space of arm-exploration of the iCub, and the skills thus developed may be different. (b) CCSA requires at least one input exploratory option. To minimize human inputs into the system, in our experiments at $t = 0$, the agent starts with only a single input exploratory option, which is a random-walk in the default abstracted-state space. However, environment or domain specific information can be used to design several input exploratory options in order to shape the resulting skills. For example, random-walk policies mapped to different sub-regions in the robot's joint space can be used.

4. Theoretical Formulation of the Learning Problem

In this section, we present a theoretical formulation of the learning problem associated with our proposed CCSA framework. We first formalize the curiosity-driven skill acquisition problem and then later in the section we present a continual extension of it.

4.1. Curiosity-driven Skill Acquisition

Given a fixed set of input exploratory options, which generate potentially high dimensional observation streams that may or may-not be unique, the objective is to

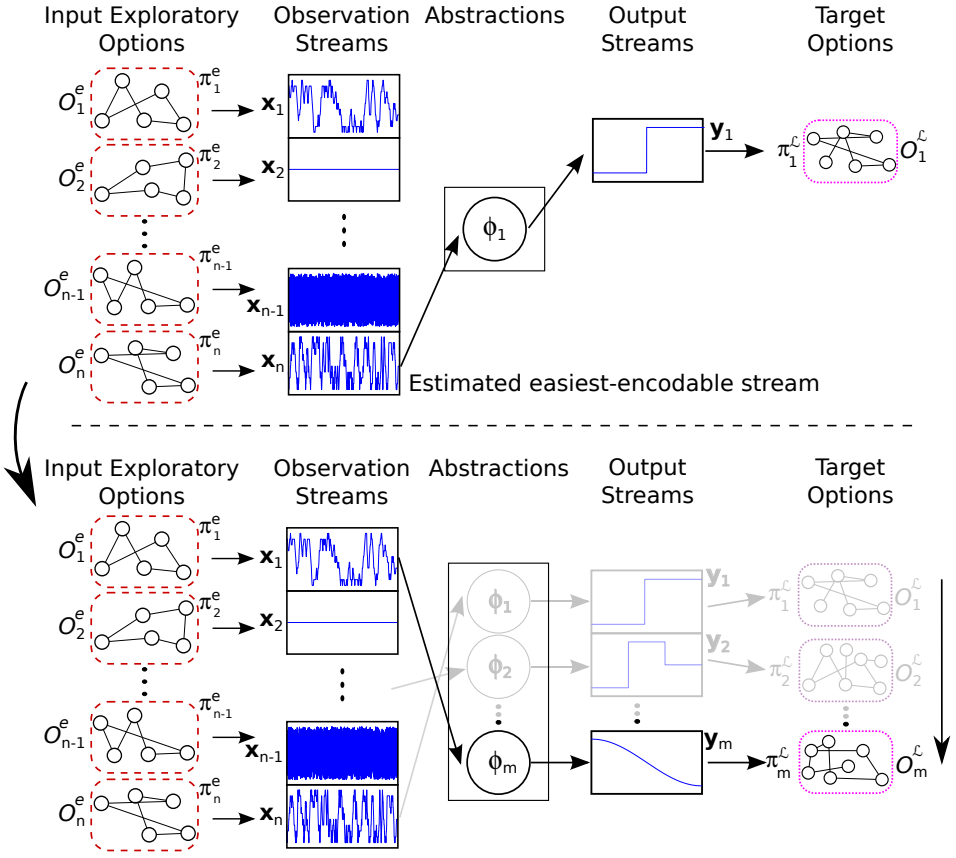

Figure 3: Curiosity-driven Skill Acquisition: Given a fixed set of input exploratory options (represented by red dashed boxes) generating n observation streams, abstractions (represented by circles) and corresponding target options (represented by pink dotted boxes) are learned sequentially in order of increasing learning difficulty. The learning process involves not just acquiring the target options, but also the sequence in which they are acquired. The top figure shows an example of the desired result after the first target option was learned. The bottom figure shows the the desired end result after all possible target options have been learned. The curved arrow indicates the temporal evolution of the learning process.

acquire a previously unknown target option corresponding to the easily-encodable observation stream. Figure 3 illustrates the learning process. The learning process iterates over the following steps:

(a) Estimate the easily-encodable yet unknown observation stream, while simultaneously learning a compact encoding (abstraction) for it.

(b) Learn an option that maximizes the statistical variance of the encoded abstraction output. The problem is formalized as follows:

4.1.1. Notation

Environment: An agent is in an environment that has a state-space $\mathcal{S}$. It can take an action $a \in \mathcal{A}$ and transition to a new state according to the transition-model (environment dynamics) $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$. The agent observes the environment state s as a high-dimensional vector, $\mathbf{x} \in \mathbb{R}^I, I \in \mathbb{N}$.

Abstraction: Let Θ denote some online abstraction-estimator that updates a feature-abstraction ϕ, where $\Theta(\mathbf{x}, \phi)$ returns an updated abstraction for an input $\mathbf{x}$. The abstraction $\phi : x \mapsto y$ maps a high-dimensional input observation stream $\mathbf{x}(t) \in \mathbb{R}^I$ to a lower-dimensional output $\mathbf{y}(t) \in \mathbb{R}^J, J \ll I, J \in \mathbb{N}$, such that $\mathbf{y}(t) = \phi(\mathbf{x}(t))$.

Abstracted-State Space: The agent's abstracted-state space $\mathcal{S}^\Phi$ contains the space spanned by the outputs $\mathbf{y}$ of all the abstractions that were previously learned using Θ.

Input Exploratory Options: The agent can execute an input set of pre-defined temporally extended action sequences, called the *exploratory option set* $\mathcal{O}^e = \{O_1^e, ..., O_n^e; n \geq 1\}$. Each exploratory option is defined as a tuple $\langle \mathcal{I}_i^e, \beta_i^e, \pi_i^e \rangle$, where $\mathcal{I}_i^e \subseteq \mathcal{S}^\Phi$ is the initiation set comprising abstracted states where the option is available, $\beta_i^e : \mathcal{S}^\Phi \rightarrow [0, 1]$ is the option termination condition, which will determine where the option terminates (e.g., some probability in each state), and $\pi_i^e : \mathcal{I}_i^e \times \mathcal{A} \rightarrow [0, 1]$ is a pre-defined *stochastic* policy, such as a random walk within the applicable state space. Each exploratory-option's policy generates an observation stream via a sensor-function $\mathcal{U}$, such as an image-sensor like a camera:

$$\mathbf{x_i}(t) = \mathcal{U}(\mathcal{P}(s, \pi_i^e(s^\Phi)))$$

where $\mathcal{P}$ is the unknown transition model of the environment, $s^\Phi \in \mathcal{I}_i^e$ is the agent's current abstracted state while executing the i^{th} exploratory option O_i^e at time t, $s \in \mathcal{S}$ is the corresponding environment state, and $\pi_i^e(s^\Phi)$ returns an action. Let $X = \{\mathbf{x_1}, ..., \mathbf{x_n}\}$ denote the set of n I-dimensional observation streams generated by the n exploratory-option's policies. At each time t however, the learning algorithm's input sample is from only *one* of the n observation-streams.

Curiosity Function: Let $\Omega : X \rightarrow [0, 1)$ denote a function indicating the speed of learning an abstraction by the abstraction-estimator Θ. Ω induces a total ordering among the observation streams making them comparable in terms of learning difficulty.[1]

Target Options: Unlike the pre-defined input exploratory-option set, a target-option set $\mathcal{O}^\mathcal{L}$ is the outcome of the learning process. A target option $O^\mathcal{L} \in \mathcal{O}^\mathcal{L}$

[1]Refer to our previous work [47, 73] for a proof on the existence of such a function and an analytical expression of Ω for IncSFA.

contains a learned abstraction ϕ_i and a learned *deterministic* policy $\pi_i^{\mathcal{L}}$. It is defined as a tuple $\langle \mathcal{I}_i^{\mathcal{L}}, \beta_i^{\mathcal{L}}, \phi_i, \pi_i^{\mathcal{L}} \rangle$. $\mathcal{I}_i^{\mathcal{L}} \subseteq (\mathcal{S}^{\Phi} \times \mathcal{S}_{\phi_i}^{\Phi})$ is the target-option's initiation set defined over the *augmented* state-space $(\mathcal{S}^{\Phi} \times \mathcal{S}_{\phi_i}^{\Phi})$, where $\mathcal{S}_{\phi_i}^{\Phi}$ denotes the space spanned by the abstraction ϕ_i's output $\mathbf{y}(t) = \phi\left(\mathbf{x_j}(t)\right), \mathbf{x_j} \in X$. β_i is the option's termination condition, and $\pi_i^{\mathcal{L}} : (\mathcal{S}^{\Phi} \times \mathcal{S}_{\phi_i}^{\Phi}) \rightarrow \mathcal{A}$ is the learned deterministic policy.

Encoded Observation Streams: Let $X^{\mathcal{O}^{\mathcal{L}}(t)}$ denote an ordered set (induced by time t) of *pre-images* of the learned abstractions outputs, $X^{\mathcal{O}^{\mathcal{L}}(t)} = \{\phi_i^{\leftarrow}\mathbf{y}_i, \forall O_i^{\mathcal{L}} \in \mathcal{O}^{\mathcal{L}}(t)\}$. $X^{\mathcal{O}^{\mathcal{L}}(t)}$ represents the set of encoded observation streams at time t.

Other Notation: $|.|$ indicates cardinality of a set, $\|.\|$ indicates Euclidean norm, $\langle . \rangle_t$ indicates averaging over time, $\langle . \rangle_t^{\tau}$ indicates windowed-average with a fixed window size τ over time. δ is a small scalar constant (≈ 0). Var[.] represents statistical variance and $\forall$ indicates $forall$.

4.1.2. Problem Statement

With the above notation, curiosity-driven skill acquisition problem can be formalized as an optimization problem with the objective that: Given a fixed set of input exploratory options $\mathcal{O}^e$, find a target-option set $\mathcal{O}^{\mathcal{L}}$, such that the number of target options learned **at any time t** is maximized:

$$\max_{\mathcal{O}^{\mathcal{L}}} \quad \left|\mathcal{O}^{\mathcal{L}}(t)\right|, \quad \forall t = 1, 2, ...$$

under the constraints,

$$\langle y_i^j \rangle_t = 0, \quad \langle (y_i^j)^2 \rangle_t = 1, \quad \forall j \in \{1, ..., J\}, \forall O_i^{\mathcal{L}} \in \mathcal{O}^{\mathcal{L}}(t) \tag{1}$$

$$\left(\begin{array}{c} \forall O_i^{\mathcal{L}} \in \mathcal{O}^{\mathcal{L}}(t), \exists j \in \{1, ..., n\}, \\ \text{and} \quad \forall O_{k \neq i}^{\mathcal{L}} \in \mathcal{O}^{\mathcal{L}}(t) \end{array} \right) : \begin{array}{l} \langle \|\Theta(\mathbf{x_j}, \phi_{\mathbf{i}}) - \phi_{\mathbf{i}}\| \rangle_{\mathbf{t}}^{\tau} \leq \delta \\ \langle \|\Theta(\mathbf{x_j}, \phi_{\mathbf{k}}) - \phi_{\mathbf{k}}\| \rangle_{\mathbf{t}}^{\tau} > \delta \end{array} \tag{2}$$

$$\Omega(\mathbf{x}_i) \leq \Omega(\mathbf{x}_j), \ \ \forall i < j \text{ and } \mathbf{x}_i, \mathbf{x}_j \in X^{\mathcal{O}^{\mathcal{L}}(t)} \tag{3}$$

$$\pi_i^{\mathcal{L}} = \arg\sup_{\pi_i} \text{Var}\left[\phi_i\left(\mathcal{U}(\mathcal{P}(s, \pi_i(s^{\Phi})))\right)\right], s^{\Phi} \in \mathcal{I}^{\mathcal{L}}, \forall O_i^{\mathcal{L}} \in \mathcal{O}^{\mathcal{L}}(t). \tag{4}$$

Constraint (1) requires that the abstraction-output components have zero mean and unit variance. This constraint enables the abstractions to be non-zero and avoids learning features for constant observation streams. Constraint (2) requires a *unique* abstraction be learned that encodes *at least* one of the input observation streams, avoiding redundancy. Constraint (3) imposes a total-ordering induced by Ω on the abstractions learned. Easier-to-learn observation streams are encoded first. And finally, Constraint (4) requires that each target-option's policy maximizes *sensitivity*, determined by the variance of the observed abstraction outputs [74]. In

the rest of the paper, we interchangeably use the word **skill** to denote a learned target option $O_i^{\mathcal{L}}$ and a **skill-set** to denote the target-option set $\mathcal{O}^{\mathcal{L}}$.

Optimal Solution: For the objective to be minimized, at any time t, the optimal solution is to learn a target option corresponding to the current *easiest but not-yet-learned* abstraction among the observation streams (to satisfy Constraints (1-3)) and a policy that maximizes the variance in the encoded abstraction output (to satisfy Constraint (4)).

However, since Ω (see Constraint 3) is not known *a priori*, it needs to be estimated online by actively exploring the input exploratory options over time. One possible approach is to find (a) an analytical expression[2] of Ω for the particular abstraction-estimator Θ and (b) an observation stream selection technique that can estimate the Ω values for each observation stream. This approach would be dependent on the abstraction-estimator used. However, our proposed framework employs an abstraction-estimator independent approach by making use of reinforcement learning to estimate the Ω values, in the form of *curiosity rewards* generated through the *learning progress* made by Θ.

4.2. Continual Curiosity-driven Skill Acquisition

In the above formulation, the agent has a fixed set of $n(\geq 1)$ input exploratory options. Therefore, the number of learnable target options is equal to the total number of learnable abstractions, which is at most equal to the number of input exploratory options:

$$\lim_{t\to\infty} \left|\mathcal{O}^{\mathcal{L}}(t)\right| \leq n. \tag{5}$$

To enable *continual learning* [29], the number of skills acquired by the agent should not necessarily be bounded and the agent needs to reuse the previously acquired skills to learn *more complex* skills. Therefore, continual curiosity-driven skill acquisition learning problem is a slightly modified version of the above formulation, such that the target options learned form a basis for *new input exploratory options*:

$$\mathcal{O}^e \leftarrow \mathcal{O}^e \cup \mathcal{F}(O^{\mathcal{L}}), \tag{6}$$

where $\mathcal{F}(.)$ denotes some functional variation of a deterministic target option to make it stochastic (exploratory). Therefore, the number of input exploratory options (n) increases whenever a new skill is acquired by the agent.

[2]Refer to our previous work [47] for an analytical expression of Ω for IncSFA.

Sub-Target Options: Constraint (4) requires that each target-option's policy maximizes variance of the observed J-dimensional abstraction outputs. However in principle, the constraint can be re-written such that only a subset of J dimensions of the abstraction can be used to learn a policy. This results in a maximum number of $2^J - 1$ learnable policies. We denote a set of target options that all share the same abstraction $\{\langle \mathcal{I}_i^{\mathcal{L}}, \beta_i^{\mathcal{L}}, \phi_i, \pi_{ij}^{\mathcal{L}} \rangle; j \leq (2^J - 1)\}$ as sub-target options. To keep it simple however, in the rest of the paper we use all the J dimensions, as presented in Constraint (4), to learn the target-option's policy and therefore limiting 1 target option for each learned abstraction.

5. Continual Curiosity-driven Skill Acquisition (CCSA) Framework

Section 3 presented an overview of our proposed framework. Here, we discuss each part of the framework in detail and also show how it addresses the learning problem formalized in Section 4.

5.1. Input Exploratory Options

As discussed in Section 4, we defined a set of input exploratory options that the agent can execute to interact with the environment. Here, we present details on how to construct these options.

The simplest exploratory-option policy is a random walk. However, we present here a more sophisticated variant that uses a form of *initial artificial curiosity*, based on *error-based rewards* [22]. This exploratory-option's policy π^e is determined by the predictability of the observations $\mathbf{x}(t)$, but can also switch to a random walk when the environment is too unpredictable.

This policy π^e has two phases. If the estimation error of *any* already learned abstraction modules for the incoming observations is lower than threshold δ, the exploratory-option's policy is learned using Least-Squares Policy Iteration Technique (LSPI; [53]), with an estimation of the transition model actively updated over the option's state-space $\mathcal{I}_i^e \subseteq \mathcal{S}^\Phi$, and an estimated reward function that rewards *high estimation errors*. Such a policy encourages the agent to explore its "unseen world" (Figure 4(a)). But if the estimation error of already learned abstraction modules is higher than the threshold δ, then the exploratory-option's policy is a random-walk over the option's state-space. Figure 4 illustrates this error seeking exploratory-option's policy. We denote this policy as LSPI-Exploration policy. When the agent selects an exploratory option O_i^e to execute, it follows the option's policy, generating an observation stream $\mathbf{x_i} = \mathcal{U}(\mathcal{P}(s, \pi_i^e(s^\Phi)))$, until the termination condition is met. To keep it general and non-specific to the environment, in all our experiments, each exploratory-option's termination condition is such that the option terminates after a fixed τ time-steps since its execution.

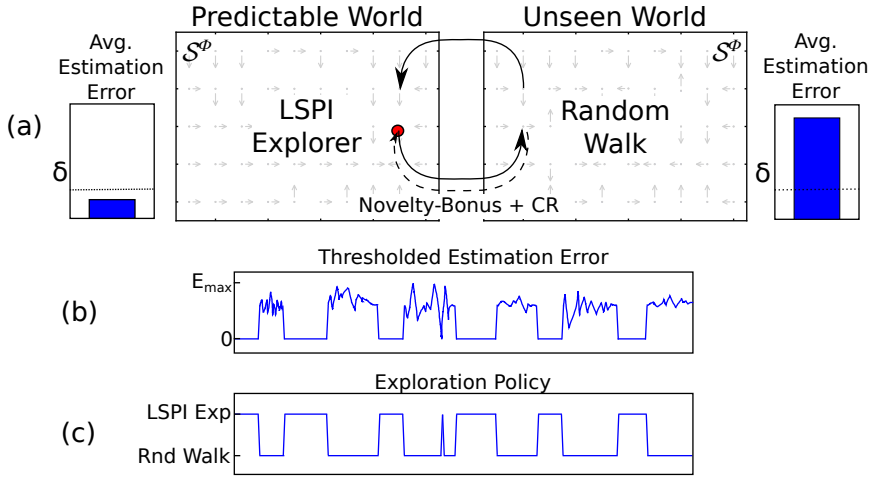

Figure 4: (a) Exploratory-option policy has two phases: If the estimation error of any already learned abstraction modules for the incoming observations is lower than threshold δ, the exploratory-option's policy is learned using Least Squares Policy Iteration (LSPI). If the estimation error is higher than the threshold then the policy is a random walk. (b) An example thresholded estimation error and the (c) corresponding exploration policy.

Setting a different input exploratory-option set would influence the skills developed by CCSA. In our experiments at $t = 0$, the agent starts with only a single exploratory option as defined above. The LSPI-Exploration policy only speeds up the agent's exploration by acting deterministically in the predictable world and randomly in unseen world. Since at $t = 0$ the world is unexplored, LSPI-Exploration policy is just a random walk in the agent's abstracted states. Environment or domain specific information can be used to design the input exploratory-option set in order to shape the resulting skills. For example, exploratory options with random-walk policies mapped to different sub-regions in the robot's joint space can be used.

5.2. Curiosity-driven Abstraction Learning: Curious Dr. MISFA

At the core of the CCSA framework is the Curiosity Driven Modular Incremental Slow Feature Analysis Algorithm (Curious Dr. MISFA; [47, 48]).[3] The order in which skills are acquired in the CCSA framework is a direct consequence of the order in which the abstractions are learned by the Curious Dr. MISFA algorithm. The input to the Curious Dr. MISFA algorithm is a set of high-dimensional observation streams $X = \{\mathbf{x_1}, ..., \mathbf{x_n} : \mathbf{x_i}(t) \in \mathbb{R}^I, I \in \mathbb{N}\}$, generated by the

[3] A *Python*-based implementation of Curious Dr. MISFA can be found at the URL: `www.idsia.ch/~kompella/codes/`.

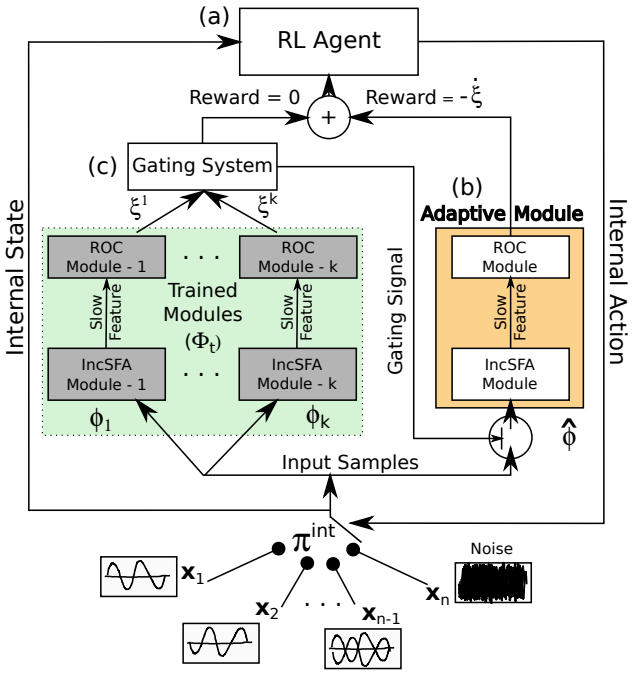

Figure 5: Architecture of Curious Dr. MISFA includes (a) a reinforcement learning agent that generates observation-stream selection policy based on intrinsic rewards, (b) an adaptive Incremental SFA coupled with Robust Online Clustering module that updates an abstraction based on the incoming observations, and (c) a gating system that prevents encoding observations that have been previously encoded.

input exploratory-option's policies. The result is a slow feature abstraction ϕ_i corresponding to the easiest yet unknown observation stream. Apart from learning the abstraction, the learning process also involves selecting the observation stream that is the easiest to encode. To this end, Curious Dr. MISFA uses reinforcement learning to learn an optimal observation-stream selection policy, based on the intrinsic rewards proportional to the progress made while learning the abstraction. In this section, we briefly review the architecture of Curious Dr. MISFA.

Figure 5 illustrates the architecture of Curious Dr. MISFA, which includes (a) a reinforcement learning (RL) agent that generates an observation-stream selection policy based on intrinsic rewards, (b) an adaptive Incremental Slow Feature Analysis coupled with Robust Online Clustering (IncSFA-ROC) module that updates an abstraction based on the incoming observations, and (c) a gating system that prevents encoding observations that have been previously encoded. The RL *agent* is within an internal environment that has a set of discrete states $\mathcal{S}^{\text{int}} = \{s_1^{\text{int}}, ..., s_n^{\text{int}}\}$, equal to the number of observation streams. In each state s_i^{int}, the agent is allowed to take only one of the two actions ($\mathcal{A}^{\text{int}}$): *stay* or *switch*. The action *stay* makes the

agent's state to be the same as the previous state, while *switch* randomly shifts the agent's state to one of the other internal states. The agent at each state s_i^{int}, receives a fixed τ time step sequence of observations ($\mathbf{x}$) of the corresponding stream $\mathbf{x}_i$.

It maintains an adaptive abstraction $\widehat{\phi} \in \mathbb{R}^{I \times J}, \widehat{\phi} \notin \Phi_t$ that updates based on the observations $\mathbf{x}$ via IncSFA-ROC abstraction-estimator. The agent receives intrinsic rewards proportional to the learning progress made by IncSFA-ROC. The observation stream selection policy $\pi^{\text{int}} : \mathcal{S}^{\text{int}} \times \mathcal{A}^{\text{int}} \rightarrow [0, 1]$ is learned from the intrinsic rewards and then used to select the observation stream for the next iteration, yielding new samples $\mathbf{x}$. These new samples, if not encodable by previously learned abstractions, are used to update the adaptive abstraction. The updated abstraction $\widehat{\phi}$ is added to the abstraction set Φ_t, when the IncSFA-ROC's estimation error falls below a low threshold δ. If and when added, a new adaptive abstraction $\widehat{\phi}$ is instantiated and the process continues. The rest of this section discusses more details on different parts of the Curious Dr. MISFA algorithm.

Abstraction-Estimator: Curious Dr. MISFA's abstraction estimator is the Incremental Slow Feature Analysis (IncSFA; [46]) coupled with a Robust Online Clustering (ROC; [75, 76]) algorithm. IncSFA is used to learn real-valued abstractions of the observations, while ROC is used to learn a discrete mapping between the abstraction outputs $\mathbf{y}$ and the agent's abstracted-state space $\mathcal{S}^{\Phi}$. In particular, each abstracted state ($s^{\Phi} \in S^{\Phi}$) has an associated ROC implementation node that estimates multiple cluster centers within the slow-feature outputs.

IncSFA is an incremental version of Slow feature analysis (SFA; [14]), which is an unsupervised learning technique that extracts features from an observation stream with the objective of maintaining an informative but slowly-changing feature response over time. SFA is concerned with the following optimization problem: Given an I-dimensional input signal $\mathbf{x}(t) = [x_1(t), ..., x_I(t)]^T$, find a set of J instantaneous real-valued functions $\mathbf{g}(\mathbf{x}) = [\mathbf{g_1}(\mathbf{x}), ..., \mathbf{g_J}(\mathbf{x})]^{\mathbf{T}}$, which together generate a J-dimensional output signal $\mathbf{y}(t) = [y_1(t), ..., y_J(t)]^T$ with $y_j(t) = g_j(\mathbf{x}(t))$, such that for each $j \in \{1, ..., J\}$

$$\Delta_j = \Delta(y_j) = \langle \dot{y}_j^2 \rangle \quad \text{is minimal} \tag{7}$$

under the constraints

$$\langle y_j \rangle = 0 \quad \text{(zero mean)}, \tag{8}$$

$$\langle y_j^2 \rangle = 1 \quad \text{(unit variance)}, \tag{9}$$

$$\forall i < j : \langle y_i y_j \rangle = 0 \quad \text{(decorrelation and order)}, \tag{10}$$

with $\langle \cdot \rangle$ and $\dot{y}$ indicating temporal averaging and the derivative of y, respectively. The goal is to find instantaneous functions g_j generating different output signals that are as *slowly varying* as possible. The decorrelation constraint (10) ensures that

different functions g_j do not code for the same features. The other constraints (8) and (9) avoid trivial constant output solutions. SFA operates on the covariance of observation derivatives, so it scales with the size of the observation vector instead of the number of states. SFA is originally realized as a batch method, requiring all data to be collected before processing. The algorithmic complexity is cubic in the input dimension I. By contrast, Incremental SFA (IncSFA) has a linear update complexity [46], and can adapt the features to new observations, achieving the slow feature objective robustly in open-ended learning environments.

ROC is a clustering algorithm similar to an incremental K-means algorithm [77] — a set of cluster centers is maintained, and with each new input, the most similar cluster center (the winner) is adapted to become more like the input. Unlike K-means, with each input it follows the adaptation step by *merging* the two most similar cluster centers, and *creating a new cluster center* at the latest input. In this way, ROC can quickly adjust to non-stationary input distributions by directly adding a new cluster for the newest input sample, which may mark the beginning of a new input process.

Estimation error and Curiosity Reward. Each ROC-Estimator node j has an associated error ξ^j. These errors are initialized to 0 and then updated whenever the node is activated by: $\xi^j(t) = \min_w \|\mathbf{y}(t) - \mathbf{v}_w\|$, where $\mathbf{y}(t)$ is the slow-feature output vector, $\mathbf{v}_w$ is the estimate of the wth cluster of the activated node and $\|.\|$ represents L^2 norm. The total estimation error is calculated as the sum of stored errors of the nodes: $\xi(t) = \sum_{j=1}^{p} \xi^j(t)$. The agent receives rewards proportional to the *derivative* of the total estimation error, which motivates it to continue executing an option that is yielding a meaningful learnable abstraction. The agent's reward function is computed at every iteration from the curiosity rewards ($\dot{\xi}$) as follows:

$$R^{\text{int}}(s^{\text{int}}, s^{\text{int}}_{-}, a^{\text{int}}) = (1-\eta)\, R^{\text{int}}(s^{\text{int}}, s^{\text{int}}_{-}, a^{\text{int}}) + \eta \sum_{t}^{t+\tau} -\dot{\xi}(t),$$

where $0 < \eta < 1$ is a discount factor, τ is the duration of the current option until its termination, $(s^{\text{int}}, s^{\text{int}}_{-}) \in \mathcal{S}^{\text{int}}$ and $a^{\text{int}} \in \{\text{stay, switch}\}$.

Observation-Stream Selection Policy. The transition-probability model $\mathcal{P}^{\text{int}}$ of the internal environment is similar to a complete graph and is given by:

$$\mathcal{P}^{\text{int}}_{i,j,stay} = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}, \quad \mathcal{P}^{\text{int}}_{i,j,switch} = \begin{cases} 0, & \text{if } i = j \\ \frac{1}{N-1}, & \text{if } i \neq j \end{cases}, \tag{11}$$

$\forall i, j \in [1, ..., N]$. Using the current updated model of the reward function R^{int} and the internal-state transition-probability model $\mathcal{P}^{\text{int}}$, we use model-based Least Squares Policy Iteration [53] to generate the agent's internal-policy $\pi^{\text{int}} : \mathcal{S}^{\text{int}} \rightarrow$

$\{\text{stay}, \text{switch}\}$ for the next iteration. The agent uses decaying ϵ-greedy strategy [5] over the internal policy to carry out an internal-action (stay or switch) for the next iteration.

Module Freezing and New Module Creation. Once the adaptive (training) module's $\widehat{\phi}$ estimation error gets lower than a threshold δ, the agent freezes and saves the IncSFA-ROC module, resets the ϵ-greedy value and starts training a new module.

Gating System and Abstraction Assignment. The already trained (frozen) modules represent our learned library of abstractions Φ_t. If a *trained* module's estimation error within an option is below the threshold δ, that option is assigned that module's abstraction and the adaptive training module $\widehat{\phi}$ will be prevented from learning via a "gating signal" (see Figure 5). There will no intrinsic reward in this case. Hence the training module $\widehat{\phi}$ will encode only data from observation streams that were not encoded earlier. Input badly encoded by all other trained modules serve to train the adaptive module.

5.3. Learning a Target Option

From the set of observations streams generated by the input exploratory options, Curious Dr. MISFA learns a slow feature abstraction (say ϕ_i) corresponding to the estimated easiest-yet-unlearned exploratory option stream (say $\mathbf{x_j}$). The abstraction's output stream $\mathbf{y_i} = \phi_i(\mathbf{x_j})$ has a zero-mean and unit-variance over time [46], and is a lower-dimensional representation of the input $\mathbf{x}_j$ (satisfies Constraint (1); see Section 4.1.2). The output values $\mathbf{y_i}(t)$ are discretized to a set of abstraction states $\mathcal{S}^{\Phi}_{\phi_i}$, which represent the newly discovered abstracted states of the agent. A deterministic target option is then constructed as follows:

Initiation Set ($\mathcal{I}^{\mathcal{L}}$): The initiation set is simply the product state-space: $\mathcal{I}^{\mathcal{L}}_i = (\mathcal{I}^e_j \times \mathcal{S}^{\Phi}_{\phi_i})$. Therefore, the option is now defined over a larger abstracted-state space that includes the newly discovered abstraction states.

Target Option Policy ($\pi^{\mathcal{L}}$): The target option policy $\pi^{\mathcal{L}}_i : \mathcal{I}^{\mathcal{L}}_i \rightarrow \mathcal{A}$ must be done in such a way as to satisfy Constraint (4). To this end, we use Model-based Least-Squares Policy Iteration Technique (LSPI; [53]) over an estimated transition and reward models. The target-option's transition model $\mathcal{P}^{O^{\mathcal{L}}_i}$ has been continually estimated from the $(s^{\Phi}, a, s^{\Phi}_{-})$ samples generated via the exploratory-option's policy π^e_j. As to estimate the reward function, the agent uses rewards proportional to the *difference of subsequent abstraction activations*:

$$r^{O^{\mathcal{L}}_i}(t) = \|\mathbf{y_i}(t) - \mathbf{y_i}(t-1)\| \tag{12}$$

$$R^{O^{\mathcal{L}}_i}(s^{\Phi}, a) = (1-\alpha) R^{O^{\mathcal{L}}_i}(s^{\Phi}, a) + \alpha r^{O^{\mathcal{L}}_i}(t), \tag{13}$$

where $\mathbf{y_i}(t) = \phi_i \left(\mathcal{U}(\mathcal{P}(s_-, \pi_j^e(s_-^\Phi)))\right)$ and $\mathbf{y_i}(t-1) = \phi_i \left(\mathcal{U}(\mathcal{P}(s, \pi_j^e(s^\Phi)))\right)$, s_- and s are the corresponding environment states, $\mathcal{P}$ is the unknown transition-model of the environment. $0 < \alpha < 1$ is a constant smoothing factor. Once the estimated transition and reward models stabilize, LSPI follows the RL objective and learns a policy $\pi_i^{\mathcal{L}}$ that maximizes the expected cumulative reward over time:

$$\pi_i^{\mathcal{L}} = \arg\sup_{\pi} \ \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r^{O_i^{\mathcal{L}}}(t) \Big| \pi, R^{O_i^{\mathcal{L}}}\right], \tag{14}$$

where γ is a discount factor close to 1. Therefore, $\pi_i^{\mathcal{L}}$ maximizes the average activation differences, which is equivalent to maximizing variance of the activations [78] (approximately[4] satisfying Constraint (4)).

Termination Condition ($\beta^{\mathcal{L}}$): The option terminates whenever the agent reaches the abstracted-state where it observes the maximum reward $\max_{(s,a)} R^{O_i^{\mathcal{L}}}$.

Each target option learned is added to the target-option set $\mathcal{O}^{\mathcal{L}}$ and the learning process iterates until all the learnable exploratory option streams are encoded. Since the expected behavior of Curious Dr. MISFA ensures that the Constraints (1-3) are satisfied [47] and the learned target-option's policy satisfies Constraint (4), the target-option set $\mathcal{O}^{\mathcal{L}}$, at any time t, therefore satisfies the required constraints.

In Section 4, we discussed an alternative to Constraint (4), where different dimensions of the learned abstraction may be used to learn multiple policies, resulting in a set of *sub-target options*. To keep it simple, we used all dimensions of an abstraction to learn a target-option's policy. However, a sub-target option set can be constructed by following the approach discussed above. Multiple reward functions can simultaneously be estimated from the (s^Φ, a, s_-^Φ) samples generated via exploratory-option's policy, and the set of sub-target options can be constructed via least-squares policy iteration in parallel.

5.4. Reusing Target Options

To make the skill acquisition open-ended and to acquire more complex skills (see Section 4.2), the learned target option $O^{\mathcal{L}}$ can be used to explore the newly discovered abstracted-state space (see Section 5.3). However, a target option may not be reused straight-away, since by definition, it differs from an exploratory option, wherein the target-option's policy is deterministic, while the exploratory-option's

[4]The error between the true and the estimated target-option policy depends on how well the transition and reward models are estimated based on the samples (s^Φ, a, s_-^Φ) generated by the exploratory-option's policy.

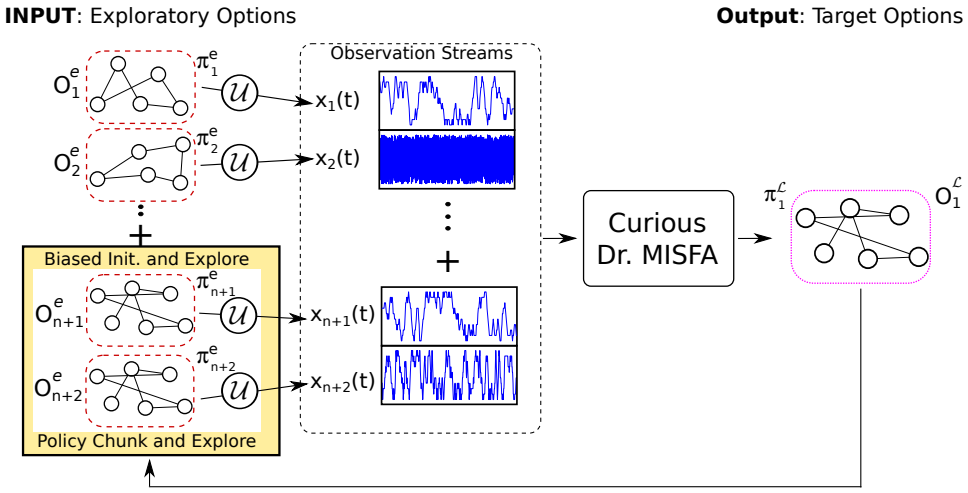

Figure 6: Reuse of the learned target options. For each target option learned (represented by pink dotted box), two new exploratory options (*Biased Initialization and Explore* and *Policy Chunk and Explore*) are added to the input exploratory-option (represented by red dashed boxes) set. Biased Initialization and Explore option biases the agent to explore first the state-action tuples where it had previously received maximum intrinsic rewards, while the Policy Chunk and Explore option executes the deterministic target-option's policy before exploration.

policy is stochastic (see Section 5.1). We construct *two* new exploratory options instead, which are based on the target option $O_i^{\mathcal{L}}$ that was learned last.

In the first option, called **policy chunk and explore**, the initiation-set is the same as that of learned target option $\mathcal{I}_{n+1}^e = \mathcal{I}_i^{\mathcal{L}}$. The policy combines the target-option's policy $\pi_i^{\mathcal{L}}$, which terminates at the state where the variance of subsequent encoded observations is highest, with the LSPI-Exploration policy described in Section 5.1. Every time this policy is initiated, the *policy-chunk* (A policy chunk is a non-adaptive frozen policy) $\pi_i^{\mathcal{L}}$ is executed, followed by the LSPI-Exploration policy. This can be beneficial if the target option terminates at a bottleneck state, after which the agent enters a "new world" of experience, within which the LSPI-Exploration policy is useful to explore.

In the second option, called **biased initialization and explore**, the exploratory-option's policy uses the normalized *value function* of the target option as an *initial reward function estimate*. This initialization biases the agent to explore first the state-action tuples where it had previously received maximum intrinsic rewards. Otherwise it is the same as the standard initial error-seeking LSPI-Exploration policy.

For each target option learned, these two exploratory options are added to the input exploratory-option set. In this way, the agent continues the process of curiosity-based skill acquisition by exploring among the new exploratory option

Algorithm 1: INT-POLICY-UPDATE ($\mathbf{x}$)

```
   // Curious Dr. MISFA Internal Policy Update
```
Abstraction-Learned $\leftarrow$ False `// Abstraction learned or not.`
$\phi \leftarrow$ Gating-System($\mathbf{x}$) `//Get the assigned abstraction.`
$\xi_{t+1} = \|\Theta(\mathbf{x}, \phi) - \phi\|$ `//Estimation Error`
if $\langle \xi_{t+1} \rangle_\tau > \delta$ **then**
  $\widehat{\phi} \leftarrow \Theta(\mathbf{x}, \widehat{\phi})$ `//Update the adaptive-abstraction`
  if $\langle \|\Theta(\mathbf{x}, \widehat{\phi}) - \widehat{\phi}\| \rangle_\tau < \delta$ **then**
    $\Phi_{t+1} \leftarrow \Phi_t \cup \widehat{\phi}$ `// Update abstraction set`
    Abstraction-Learned $\leftarrow$ True
  end
end
$R^{\text{int}}_{t+1} \leftarrow$ UpdateReward ($\dot{\xi}_{t+1}$) `//Update the int. reward func.`
$\pi^{\text{int}}_{t+1} \leftarrow$ Model-LSPI ($\mathcal{P}^{\text{int}}, R^{\text{int}}_{t+1}$) `//Update int. policy`
$\pi^{\text{int}}_{t+1} \leftarrow \epsilon$-greedy ($\pi^{\text{int}}_{t+1}$) `//Exploration-exploitation tradeoff`
return (π^{int}_{t+1}, Abstraction-Learned)

set to discover unknown regularities. A complex skill $O^{\mathcal{L}}_k = \langle \mathcal{I}^{\mathcal{L}}_k, \beta^{\mathcal{L}}_k, \phi_k, \pi^{\mathcal{L}}_k \rangle$ can be learned as a consequence of chaining multiple skills that were learned earlier.

5.5. Pseudocode

The entire learning process involves determining three policies:

1. π^e: Exploratory-option's stochastic policy that is determined (see Section 5.1) to generate high-dimensional observations.

2. π^{int}: An internal policy that is learned (see Section 5.2) to determine for which exploratory option O^e to encode a slow feature abstraction.

3. $\pi^{\mathcal{L}}$: Target-option's deterministic policy that is learned (see Section 5.3) to maximize variation in the slow feature abstraction output.

The resultant target options (skills) are stored and reused as discussed above to facilitate open-ended continual learning. Algorithms 1 and 2 summarize the entire learning process.[5]

[5] *Python*-based code excerpts can be found at the URL: `www.idsia.ch/~kompella/codes/`.

Algorithm 2: CONTINUAL CURIOSITY-DRIVEN SKILL ACQUISITION (CCSA)

$\Phi_0 \leftarrow \{\}$, $\pi_0 \leftarrow$ RANDOM (), $\widehat{\phi} \leftarrow 0$, Abstraction-Learned $\leftarrow$ False
for $t \leftarrow 0$ *to* ∞ **do**
  $s^{\text{int}} \leftarrow$ current internal state, $a^{\text{int}} \leftarrow$ action selected by π_t^{int} in state s^{int}
  Take action a^{int}, observe next internal state $s_-^{\text{int}}(= i)$
  `// Execute the exploratory option` O_i^e
  while *not* $\beta_i^e(t)$ **do**
    $s^{\Phi} \leftarrow$ current abstracted-state, $a \leftarrow$ action selected by π_i^e in state s^{Φ}
    Take action a, observe next abstracted-state s_-^{Φ} and the sample $\mathbf{x}$
    if *not Abstraction-Learned* **then**
      `// Internal Policy Update`
      (π_{t+1}^{int}, Abstraction-Learned) = Int-Policy-Update ($\mathbf{x}$)
    else
      `// Learn target option`
      $\pi_{t+1}^{\text{int}} \leftarrow \pi_t^{\text{int}}$, $R_{\text{prev}} \leftarrow R^{O^{\mathcal{L}}}$, $P_{\text{prev}} \leftarrow P^{O^{\mathcal{L}}}$
      $R^{O^{\mathcal{L}}}(s^{\Phi}, a) = (1-\alpha)R^{O^{\mathcal{L}}}(s^{\Phi}, a) + \alpha(\|\mathbf{y_i}(t) - \mathbf{y_i}(t-1)\|)$
      $P^{O^{\mathcal{L}}}(s^{\Phi}, a, s_-^{\Phi}) = (1-\alpha)P^{O^{\mathcal{L}}}(s^{\Phi}, a, s_-^{\Phi}) + \alpha$
      if ($\|R^{O^{\mathcal{L}}} - R_{prev}\| < \delta$ *and* $\|P^{O^{\mathcal{L}}} - P_{prev}\| < \delta$) **then**
        $\pi^{\mathcal{L}} \leftarrow$ LSPI-Model($P^{O^{\mathcal{L}}}$, $R^{O^{\mathcal{L}}}$)
        $O^{\mathcal{L}} = \langle \mathcal{I}^{\mathcal{L}}, \beta^{\mathcal{L}}, \widehat{\phi}, \pi^{\mathcal{L}} \rangle$ `// Construct target option`
        $\mathcal{O}^{\mathcal{L}} \leftarrow \mathcal{O}^{\mathcal{L}} \cup O^{\mathcal{L}}$ `// Add to target-option set`
        `// Construct two new exploratory options`
        $\mathcal{O}^e \leftarrow \mathcal{O}^e \cup$ Biased-Init-Explore($O^{\mathcal{L}}$)
        $\mathcal{O}^e \leftarrow \mathcal{O}^e \cup$ Policy-Chunk-Explore($O^{\mathcal{L}}$)
        $\widehat{\phi} \leftarrow 0$, Abstraction-Learned $\leftarrow$ False `// Reset`
      end
    end
  end
end

6. Experimental Results

We present here experimental results that focus on continual-learning of skills using an iCub humanoid platform. More studies on the types of representations learned by the IncSFA algorithm and curiosity-based abstraction learning with Curious Dr. MISFA can be found elsewhere [47, 48, 46, 68]. The results here are the **first** in which a humanoid robot such as an iCub, learns a repertoire of skills

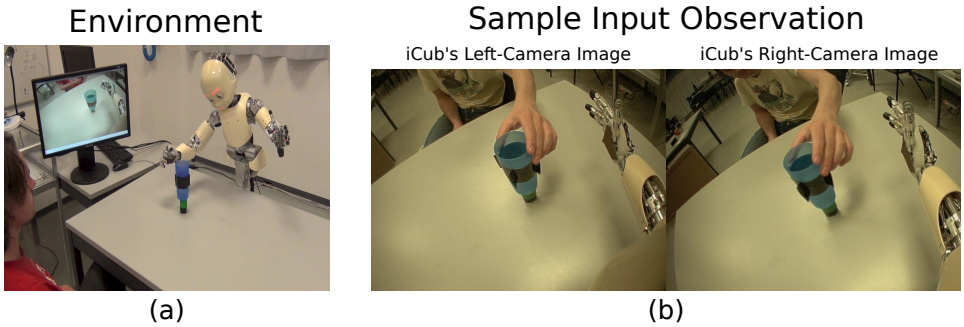

Figure 7: (a) An iCub robot is placed next to a table, with an object (a plastic cup) in reach of its right arm and within its field-of-view. (b) Sample input images captured from both left and right iCub camera-eyes are an input to the algorithm.

from *raw-pixel data* in an *online* manner, driven by its own *curiosity*, starting with low-level joint kinematic maps.[6]

Learning a skill-set largely depends on the environment that the robot is in. For the sake of developing specific types of skills such as toppling an object, grasping, etc., we pre-selected a safe environment for the iCub to explore, yet the iCub is mostly unaware of the environment properties.

Environment: Our iCub robot is placed next to a table, with an object (a plastic cup) in reach of its right arm and within its field-of-view (Figure 7(a)). The cup topples over upon contact, and the resulting images after toppling are predictable. There is a human experimenter present, who monitors the robot's safety and replaces the cup in its original position after it is toppled. The iCub does not "know" that the plastic-cup and the experimenter exist. It continually observes the gray-scale pixel values from the high-dimensional images (75×100) captured by the left and right camera eyes (Figure 7(b)). In addition to the experimenter and the cup, it also cannot recognize its own moving hand in the incoming image stream, as shown in the Figure 7(b).

Task-Relevant Roadmap We do not induce exploration at the level of joint angles, due to the complexity of the robot's joint space. Instead we give the robot a map of poses *a priori*. This compressed actuator joint-space representation is called a Task-Relevant Roadmap (TRM; [50]). This map contains a family of iCub postures that adhere to relevant constraints. The TRM is grown offline by repeatedly optimizing cost-functions that represent the constraints, using a Natural Evolution Strategies (NES; [79]) algorithm, such that the task-space is covered. This allows

[6] A video for these experiment can be found at URL: `http://www.youtube.com/watch?v=OTqdXbTEZpE`

us to deal with complex cost-functions and the full 41 degrees-of-freedom of the iCub's upper body. The constraints used: (a) the iCub's hand is positioned on a 2D plane parallel to the table while keeping its palm oriented horizontally, (b) the left hand is kept within a certain region to keep it out of the way, and (c) the head is pointed towards the table. The task-space of the TRM comprises the x and y position of the hand, which forms the initial discretized 10×5 abstracted-state space $\mathcal{S}^{\Phi} = \mathcal{S}^{\Phi}_x \times \mathcal{S}^{\Phi}_y$. The action space contains 6 actions: move *North*, *East*, *South*, *West*, *Hand-close* and *Hand-open*.

Because the full body is used, the movements look more dynamic, but as a consequence, the head moves around and looks at the table from different directions, making the task a bit more difficult. Even so, IncSFA still finds the resulting regularities in the raw camera observation stream, and the skill learner continues to learn upon these regularities, without any external rewards.

Experiment parameters: We use a fixed parameter setting for the entire experiment.

IncSFA Algorithm: IncSFA has two learning update rules [46]: Candid-Covariance free Incremental Principal Component Analysis (CCIPCA; [80]) for normalizing the input and Minor Component Analysis (MCA; [81]) for extracting slow features. For CCIPCA, we use learning rates $1/t$ with amnesic parameter 0.4, while for MCA the learning rate is set to 0.01. CCIPCA does variable size dimension reduction by calculating how many eigenvalues would be needed to keep 99% of the input variance — typically this was between $5-10$ — so the 7500 pixels could be effectively reduced to only about 10 dimensions. The output dimension is set to 1, therefore, we use only the first IncSFA feature as an abstraction. However, more number of features can be used if desired.

Robust Online Clustering (ROC) Algorithm: ROC algorithm maps slow-feature outputs to abstracted states (see Section 5.2). Each clustering implementation has its maximum number of clusters set to $N^{max} = 3$, such that it can encode multiple slow feature values for each abstracted state. Higher values can be used, however, very high values may lead to spurious clusters. The estimation error threshold, below which the current module is saved and a new module is created, is set to a low value $\delta = 0.3$. The amnesic parameter is set to $\beta^{amn} = 0.01$. Higher values will make ROC adapt faster to the new data, however at the cost of being less stable.

Curious Dr. MISFA's Internal Reinforcement Learner: To balance between exploration and exploitation, ϵ-greedy strategy is used (see Section 5.2). The initial ϵ-greedy value is set to 1.0 (1.0 for pure exploration, 0.0 for pure exploitation), with a 0.995 decay multiplier. The window-averaging time constant is set to $\tau = 20$, that is, 20 sample images are used to compute the window-averaged progress error ξ and the corresponding curiosity-reward (see Section 5.2).

Target-option's Reinforcement Learner: Slow features abstractions have unit-variance and are typically in the range of $(-1.5, 1.5)$ [46]. Since in our experiments we are expecting step-like slow features, to keep it simple, each abstraction-output values are discretized to either $(-1, 1)$, therefore into two $|\mathcal{S}^{\phi_i}| = 2$ abstracted states.

Experiment Initialization: The iCub's abstracted-state space ($\mathcal{S}^{\Phi}$) at $t = 0$ is a 10×5 grid found using TRM. To minimize human input into the system, the input exploratory-option set ($\mathcal{O}^e$) has only one exploratory option to begin with (as defined in Section 5.1): $\mathcal{O}^e = \{O_1^e\}$, which is a random-walk in the iCub's abstracted-state space. However, one may pre-define multiple input exploratory options, which could lead to a different result. The exploratory option terminates after $\tau = 20$ time steps since its execution. The internal state-space at $t = 0$ is $\mathcal{S}^{\text{int}} = \{s_1^{\text{int}}\}$, where s_1^{int} corresponds to the exploratory option O_1^e. The plastic cup is roughly placed around $(2, 2)$ grid-point on the table.

6.1. iCub Learns to Topple the Cup

The iCub starts the experiment without any learned modules, so the exploratory-option's policy π_1^e is a random-walk over the abstracted state space $\mathcal{S}^{\Phi}$ (see Section 5.4). It explores by taking one of the six actions: *North*, *East*, *South*, *West*, *Hand-close* and *Hand-open* and grabs high-dimensional images from its camera-eyes. The exploration causes the outstretched hand to eventually displace or topple the plastic-cup placed on the table. It continues to explore and after an arbitrary amount of time-steps the experimenter replaces the cup to its original position. After every τ time-steps, the currently executing option terminates. Since there is only one exploratory option, the iCub re-executes the same option. Figure 8(a) shows a sample input image stream of only the left-camera.[7]

Figure 8(b) shows the developing IncSFA output over the algorithm execution time, since the IncSFA abstraction was created. The outcome of IncSFA abstraction learning is a step-like function, which when discretized, indicates the pose of the cup (toppled vs non-toppled). Figure 8(c) shows the ROC estimation error (blue solid line) and an Expected Moving Average (EMA) of the error (green dashed line) over the algorithm execution time. As the process continues, the error eventually drops below the threshold $\delta = 0.3$ and the abstraction module ϕ_1 is saved. Figure 9(a) shows the ROC cluster centers that map the feature outputs $(\mathbf{y})$ to each of the 10×5 abstracted states. There are two well separated clusters each representing the state of the plastic-cup.

[7] We, however, used both the left and right camera images as an input observation by concatenating them.

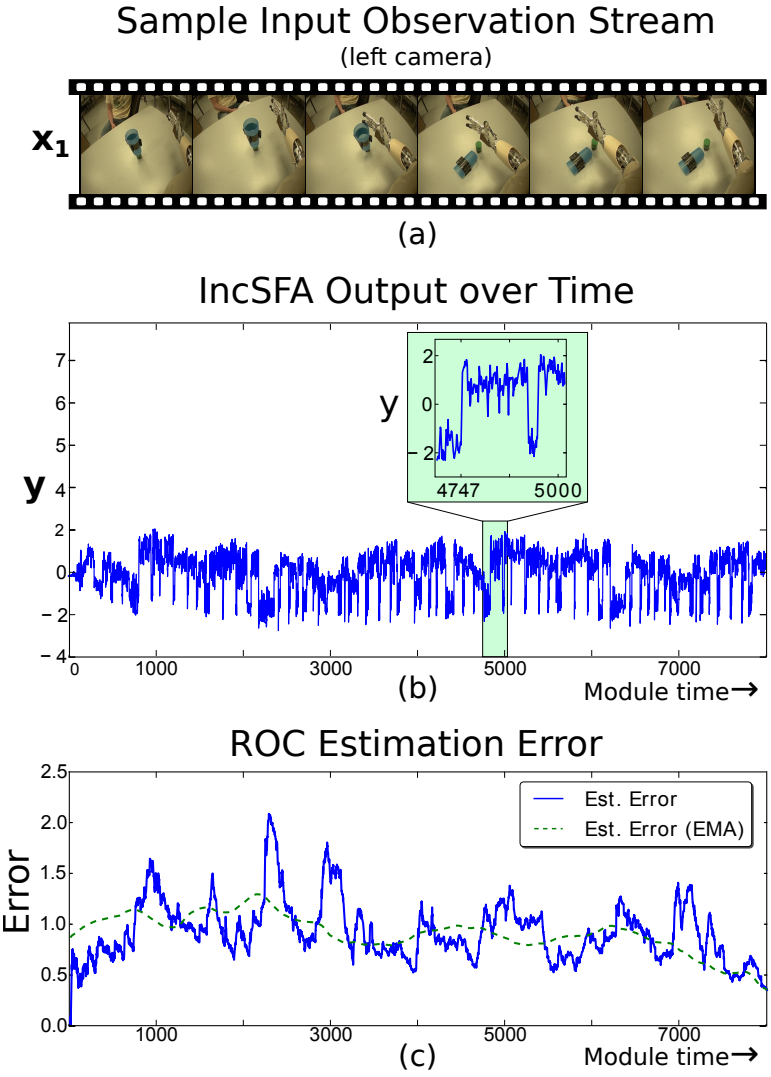

Figure 8: (a) A sample image stream of the iCub's left-eye camera showing the topple event. (b) Developing IncSFA abstraction output over algorithm execution time, since it was created. The result is a step-like function encoding the topple event. (c) ROC estimation error over algorithm execution time. The estimation error eventually drops below the threshold ($\delta = 0.3$), after which the abstraction is saved.

Immediately after the abstraction is saved, the cluster centers are discretized (Red and yellow colors indicate the discretized feature states $\mathcal{S}^{\Phi}_{\phi_1}$ in Figure 9(a)), the transition model (represented by the blue lines in Figure 9(a)) and reward model of $O^{\mathcal{L}}_1$ are learned, followed by a corresponding target-option's policy $\pi^{\mathcal{L}}_1$ as discussed in Section 5.3. Figure 9(b) shows a part of the learned policy $\pi^{\mathcal{L}}_1$ before

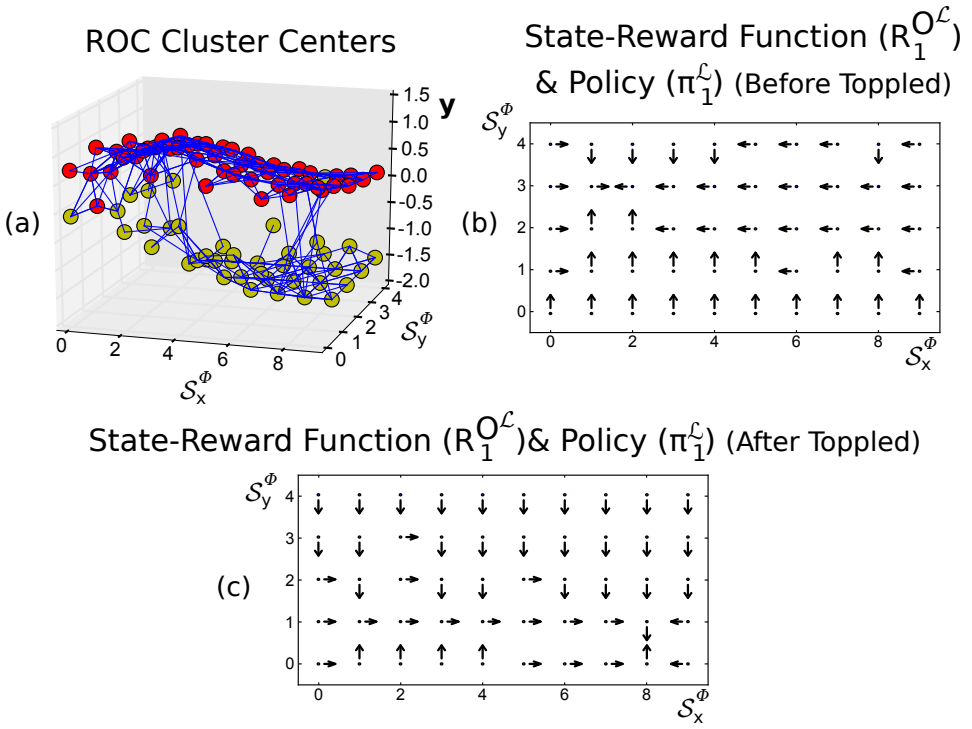

Figure 9: (a) The resultant ROC cluster centers, which map the abstraction outputs to the abstracted-state space (in this case the X and Y grid locations of the iCub's hand). Red and yellow colors indicate the discretized feature states $\mathcal{S}^\Phi_{\phi_1}$. Blue lines connecting the cluster centers illustrate the learned transition model of the new abstracted-state space. (b) Part of the learned target-option's policy before the cup is toppled. The arrows indicate the optimal action to be taken at each grid-location (s_x^Φ, s_y^Φ) of the iCub's hand. They direct the iCub's hand to the grid point $(1, 3)$, which will make the iCub topple the cup placed at $(2, 2)$. (c) Part of the learned target-option's policy after the cup is toppled. They direct the iCub's hand to move to the right. This is a result of the experimenter replacing the cup only when the iCub has moved its hand away from the $(2, 2)$ grid location.

the cup is toppled. The arrows indicate the optimal action to be taken at each grid-location of the iCub's hand. They direct the iCub's hand to the grid point $(1, 3)$, which will make the iCub topple the cup placed at $(2, 2)$. Figure 9(c) shows the part of the policy after the cup has been toppled. The policy directs the iCub's hand to move towards *east*. This is because, during the experiment the experimenter happened to replace the cup only when the iCub's hand is around far east. We label the learned target option $O_1^\mathcal{L}$, for the given environment, as a "Topple" skill.

6.2. iCub Learns to Grasp the Cup

The iCub continues its learning process by reusing the learned topple skill to construct two additional exploratory options as discussed in Section 5.4. One in

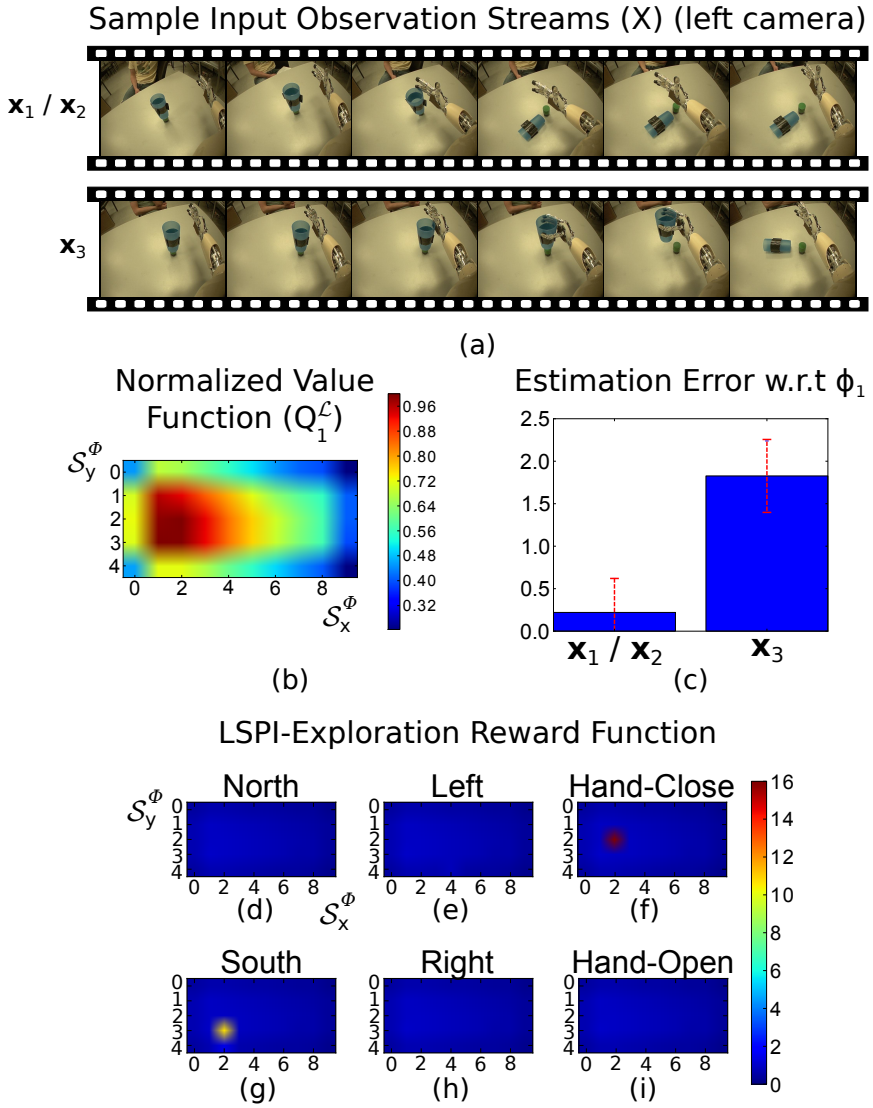

Figure 10: (a) Sample iCub's left-eye camera images corresponding to the three input exploratory options. $\mathbf{x}_1$ and $\mathbf{x}_2$ correspond to the original and the *policy chunk & explore* exploratory option respectively, while $\mathbf{x}_3$ corresponds to the *biased init. & explore* exploratory option. (b) Normalized value function of the previously learned target option (topple). It is used for reward-initialization in the *biased init. & explore* exploratory option. (c) Estimation error of the learned topple abstraction module (ϕ_1) for each of the three observation-streams. (d)-(i) LSPI-Exploration reward function estimated using the novelty (& curiosity) signal. The Hand-Close action at $(2, 2)$ has the maximum reward value due to the novel grasp event.

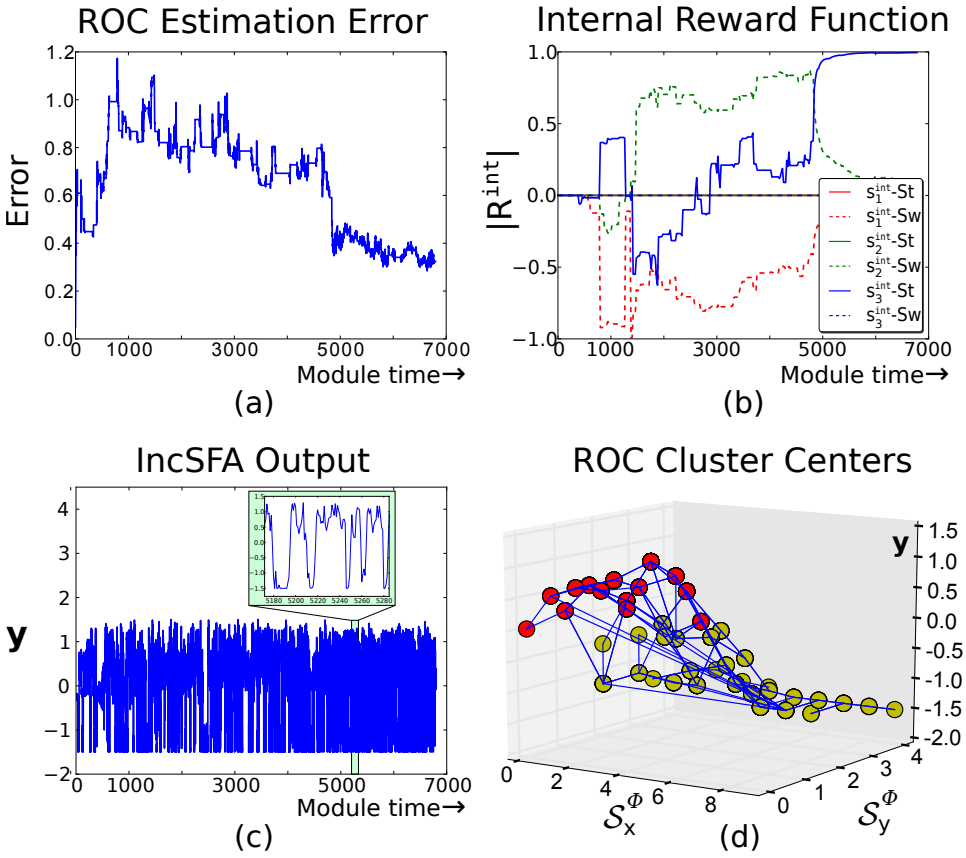

Figure 11: (a) ROC estimation error of the current adaptive-module that is encoding the new regularities. (b) Normalized internal-reward function of Curious Dr. MISFA. The action *stay* in the state corresponding to the exploratory option 3 (shown as s_3^{int}-St) is most rewarding due to the learning progress made by the IncSFA-ROC module for the grasp-event. (c) IncSFA output over execution time, since it was created. (d) Resultant ROC cluster centers mapping the IncSFA output w.r.t. the abstracted-state space. Note that the abstracted states corresponding to the learned topple abstraction $\mathcal{S}_{\phi_1}^{\Phi}$ are not shown here, since the grasp abstraction outputs are uncorrelated to the topple abstraction and it is difficult to illustrate a 4-D plot. Red and yellow colors indicate the discretized states $\mathcal{S}_{\phi_2}^{\Phi}$ and the blue lines illustrate the learned transition model.

which the topple policy (Figure 9(b)) is executed prior to the LSPI-Exploration policy and the other, where the normalized value function (Figure 10(b)) is used to initialize the reward-function of the LSPI-Explorer. Let O_2^e and O_3^e denote these two exploratory options respectively. Therefore, including the original exploratory option O_1^e, a total of 3 exploratory options are an input to CCSA.

Initially, the system explores by executing each of the options until termination, *i.e.*, after τ time steps. When it selects either O_1^e or O_2^e, the cup gets toppled in the

process (Figure 10(a)-Top) and since there already exists a learned abstraction ϕ_1 that encodes the toppling outcome, it receives no internal reward for executing these options because of the gating system (see 5.2). This is also the case in the beginning while executing O_2^e, because the LSPI-Exploration policy initially causes the iCub to topple the cup, yielding no rewards. The initialized values corresponding to the visited state-action tuples soon vanish and the iCub then explores the neighboring state action pairs. Eventually, as a result of the biased exploration, in a few algorithm iterations the iCub ends up grasping the cup (Figure 10(a)-Bottom). This gives rise to a high estimation error because of the novelty of the event (Figure 10(c)). Figures 10(d)-(i) show the state-action LSPI-Exploration reward function after a few time steps. The hand-close action at $(2, 2)$ generates the most novel event. This results in a LSPI-Exploration policy that increases the number of successful grasp trials (77 out of 91 total attempts, with most of the unsuccessful trials in the beginning) when the exploratory option O_3^e is executed.

Now, upon executing option O_3^e, the adaptive abstraction $\hat{\phi}$ begins to make progress by encoding samples corresponding to the observation stream $\mathbf{x_3}$. After a few algorithm iterations, the agent finds that the action *stay* at the internal state s_3^{int} corresponding to the O_3^e is rewarding due to the progress made by IncSFA and the ROC estimator (Figure 11(a)). Figure 11(b) shows the normalized internal reward function of Curious Dr. MISFA over algorithm iterations, since the new adaptive module was created. The internal policy π^{int} quickly converges to select and execute the option O_3^e to receive more observations. When the estimation error drops below the threshold ($\delta = 0.3$), it saves the module $\phi_2 = \hat{\phi}$. Figure 11(c) shows the IncSFA output over the time since the new module was created. Figure 11(d) shows the learned cluster centers mapping the slow-feature output to the abstracted-state space. Note that the abstracted states corresponding to the learned topple abstraction $\mathcal{S}_{\phi_1}^{\Phi}$ and not are shown in Figure 11(d), because the grasp abstraction outputs are uncorrelated to the topple abstraction and it is difficult to illustrate a 4-D plot. The iCub then begins to learn the target policy $\pi_2^{\mathcal{L}}$ by learning the target-option's transition and reward model. Figure 12(a)-(f) show the target-option's state-action reward model developed after 8000 observation samples (module time=8000). And finally, Figure 12(g) shows the corresponding skill learned, *i.e.*, to perform a Hand-Close at $(2, 2)$ (the anti clockwise circular arrow represents the Hand Close action).

This experiment demonstrated how the iCub reused the knowledge gained by the topple skill to learn a subsequent skill labeled as "Grasp". The grasp skill includes an abstraction to represent whether the cup has been successfully grasped-or-not and a policy that directs the iCub's hand to move to $(2, 2)$ and then to close its hand.

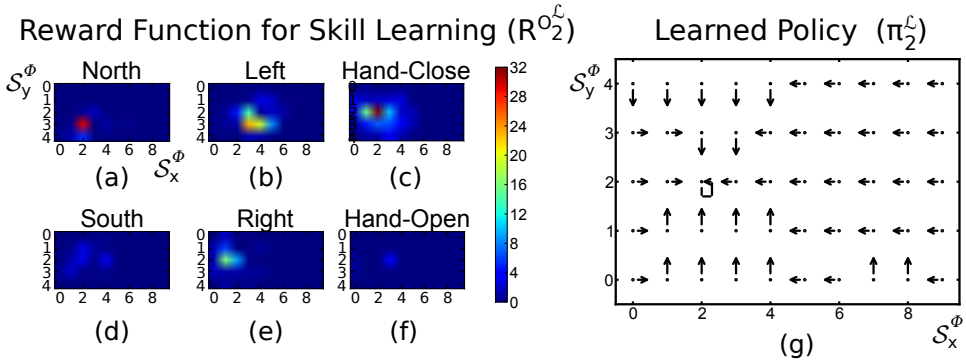

Figure 12: (a)-(f) Estimated reward-function of the new abstracted-state space that is used to learn the target-option's policy. The hand-close action at $(2, 2)$ receives the maximum reward as it produces a maximum variation in the slow-feature output (from ≈ -1.5 to 1.5). (g) Learned target-option's policy representing the grasp skill. The arrows indicate the optimal actions to be taken at each grid-location (s_x^{Φ}, s_y^{Φ}). The circular arrow represents the hand-close action. The policy directs the iCub's hand to move to $(2, 2)$ and then to close its hand, which should result in a successful grasp.

6.3. iCub Learns to Pick and Place the Cup at the Desired Location

We present here an experiment to demonstrate the utility of intrinsic motivation in solving a subsequent external objective. The iCub is in a similar environment as discussed above. However, it is given an external reward if it picks the plastic cup and places (drops) it at a desired location (at any of the following grid locations (s_x^{Φ}, s_y^{Φ}): $(6, 2), (6, 3), (6, 1), (5, 2), (7, 2)$). The agent with no intrinsic motivation finds the reward almost inaccessible via random exploration over its abstracted-state space $\mathcal{S}^{\Phi}$, because the probability of a successful trial is low.[8] $(\approx 10^{-5})$ However, a curiosity driven iCub greatly improves this by learning to pick/grasp the cup by itself and then reusing the skill to access the reward.

Starting from the 10×5 abstracted-state space found via TRM, the iCub learns to topple and then grasp as discussed in the previous sections. The process continues and it adds two more exploratory options (O_4^e, O_5^e) corresponding to the grasp skill as discussed in Section 5.4. The biased initialization and explore option O_4^e results in the iCub dropping the cup close to where it has picked it up. Since it doesn't get any reward in this case, the initialized values to the visited state-actions tuples vanish and it explores the neighboring state-action tuples. This option will take a long time before it can execute the desired state-action tuple to drop the cup. The policy chunk and explore option O_5^e, however, first executes the grasp policy and then randomly explores until it receives some novelty or curiosity re-

[8] The probability of a successful pick = 1/300, probability of a drop given a successful pick = 1/300 * 1/60.

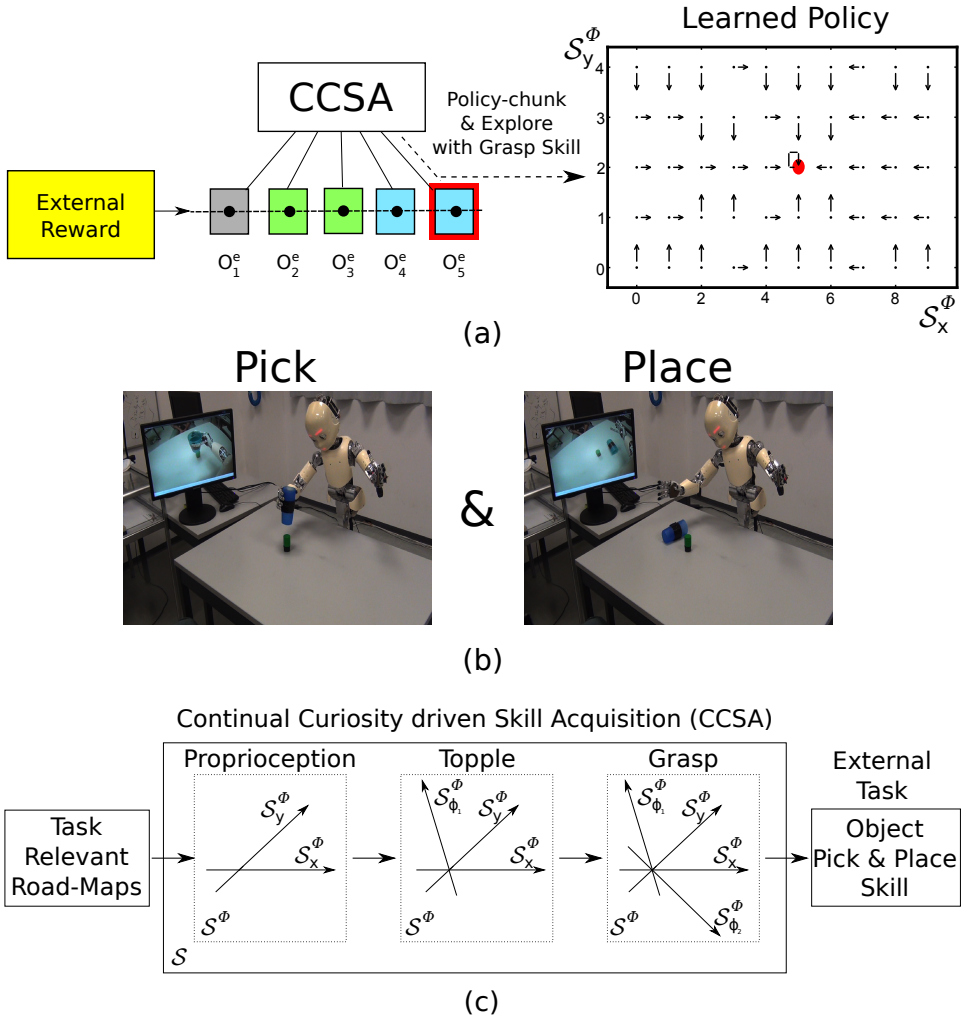

Figure 13: (a) CCSA now has 5 exploratory options as an input. Among the 5 options, only the *policy chunk & explore* corresponding to the grasp skill makes it easier for the iCub to access the external-reward present for placing the cup at the desired grid locations. This results in a policy – to place the cup in the desired location (the clockwise circular arrow represents the Hand-Open action). (b) Bird's eye view of the iCub demonstrating the pick & place skill. (b) Figure shows the increasing dimensions in the agent's abstracted-state space with every new abstraction learned. This experiment demonstrates how CCSA enables the iCub to reuse the grasp skill, which was previously learned via intrinsic motivation, on learning to pick & place the cup to a desired location.

ward. When, it drops the cup in one of the desired states while exploring, it gets an external reward, which results in a LSPI-Exploration policy that executes the rewarding behavior. Curious Dr. MISFA eventually finds the internal action *stay*

at the internal-state s_5^{int} corresponding to the option O_5^e most rewarding. As soon as the experimenter replaces the cup, the iCub repeats the pick and place behavior until the external reward is removed.

This experiment demonstrated how CCSA enabled the iCub to reuse the grasp skill, which was previously learned via intrinsic motivation, on learning to pick and place the cup to a desired location. Note that in our experiments, a human experimenter unknown to the robot, acted as a part of the environment to speed up the learning process. Without the experimenter, the robot might not have acquired the same set of skills, instead it might have learned to push the object (refer to our previous work [48] for an experiment where a simulated iCub learns a slow feature abstraction that encodes a push).

7. Discussion

While, much of the research in humanoid robot learning has been based upon human demonstrations, human-given task-descriptions, or pre-processed inputs, CCSA makes an important step towards combining several aspects needed to develop an online, continual curiosity-driven humanoid robotic agent. In the following we briefly discuss these aspects along with current limitations of our framework and insights for future work:

Raw High-Dimensional Information Processing. CCSA uses a *linear* IncSFA algorithm updated online directly from raw-pixels to encode abstractions that lead to acquiring skills. To learn more complex skills however, CCSA might benefit from extracting non-linearities in the video inputs. Hierarchical extensions of IncSFA (H-IncSFA) over an *expanded input* in quadratic space [82] may remedy this. We plan to combine non-linear hierarchical structures to further improve the quality of the abstractions learned.

Invariant Skills. The skill labeled "grasp" in our experiments actually represents "grasp the cylindrical cup from the particular location in the given environment, invariant to the experimenter's actions and the iCub's head/body movements". The invariance picked up by the skills acquired in our system largely depend on the invariance learned by IncSFA from the observations sensed by the exploring iCub. Refer to our previous work [47, 46] for more details on invariance extracted by IncSFA. In our experiments however, if the human experimenter had replaced the cup at different locations whenever the cup was toppled or dropped, we expect IncSFA to learn an abstraction that encodes whether the cup has been grasped-or-not invariant to the cup's position (because the events are uncorrelated). This would result in a "grasp skill" that is invariant to the cup's position.

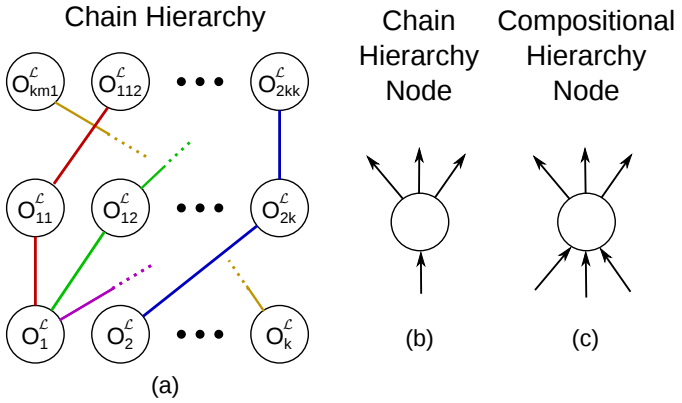

Figure 14: (a) The higher-order complex skills acquired using CCSA, are in the form of a chain-like hierarchy. There exists only a single chain-link (shown as a unique color) connecting higher-order to lower-order skills. This is because, a target option in CCSA is learned using observations only from one of the exploratory options. (b) An illustration of a node in a chain hierarchy. Each node has only a single input but can act as an input to many nodes. (c) Whereas, a node in a compositional hierarchy can have multiple inputs.

Continual Learning. CCSA uses previously acquired knowledge in the form of biased explorations or policy-chunks, to learn more complex skills. This facilitates continual learning of skills. A previously acquired skill may be refined or adapted to suit to changing environments. For example, in our experiments, if the cup's position has changed after acquiring the grasp skill, the *biased init. and explore* exploratory-option corresponding to the grasp skill can speed up learning a new skill to grasp the cup from the new position. However, the old skills are still retained and reused if the cup's position is changed back to its original position. The complex skills acquired using CCSA are in the form of a chain-like hierarchy (Figure 14(a)), *i.e.*, there exists only a single chain-link connecting higher-order to lower-order skills. This is because, a target option in CCSA is learned using observations only from one of the exploratory options (see Section 4). Each node in the chain-like hierarchy has only a single input but can act as an input to many nodes (Figure 14(b)). Whereas, a node in a general compositional hierarchy (Figure 14(c)) can have multiple inputs. One way to achieve compositional hierarchy in CCSA is to add the learned target options to the primitive action set $\mathcal{A} = \{\text{North, East, South, West, Hand-close and Hand-open}\}$.

Environment Openness. CCSA could benefit from a larger set of pre-defined input exploratory options. However, to minimize human inputs into the system, in our experiments the iCub starts with only a single exploratory option (random-walk) and autonomously adds more exploratory options derived from the learned target options. Since CCSA acts directly on raw-pixels, no prior calibration of the

robot cameras are required. Algorithm parameters are intuitive to tune. Refer to our previous work [47, 48, 46, 68] for a detailed description on tuning IncSFA and Curious Dr. MISFA algorithms. Therefore, CCSA can be used in different environments (and different humanoid robots) without making any design changes to the learning algorithm. On the motor end, we used a kinematic map that transforms the 41 degrees-of-freedom of the iCub joint configurations to 2D positions of its hand parallel to the table. For more complex manipulations, which are required for handling complicated objects, higher dimensional kinematic-maps could be used [50]. As our future work, we plan on using different approaches to tackle easier and safer manipulation with the iCub.

Quality of Skills Acquired. We presented formally the underlying learning problem as a constrained optimization problem. The objective function can be used as a metric to tune different parameters of the method. However, the metric does not sufficiently evaluate the quality of skills acquired. One major factor is the type of the abstraction-estimator used. For example, a method that uses a simpler abstraction learning algorithm may acquire a large number of skills, which could be functionally equivalent to acquiring a single skill of a more discriminative abstraction estimator. Therefore, evaluating different task-unrelated intrinsically-motivated (IM) approaches without providing an external goal is an ill-posed problem. As our future work, we plan to build realistic, task-independent, skill-acquisition benchmarks with hidden external tasks to evaluate multiple IM approaches.

Scalability. For each target option acquired by CCSA, the number of input exploratory options increases by a value of two (See Section 5.4). Observations from previously encoded exploratory options are automatically filtered out due to the gating system of Curious Dr. MISFA. Therefore, for each target option acquired, the number of unknown exploratory options increases by a value of only one. Hence, the space of input exploratory options scales linearly with respect to the number of skills acquired.

Sensor Fusion. And finally, CCSA uses only visual inputs from the onboard cameras and joint angles of the iCub. A humanoid-robot's actions can be improved however, by using different sensory modalities such as tactile and audio in addition to the visual inputs. This should be straightforward addition to CCSA, since IncSFA is agnostic to the modality of sensory information. The raw inputs of different modalities can be concatenated as a single input and fed to the IncSFA algorithm, without causing too much computational overhead (since IncSFA has a linear update complexity [46]). Related work on combining sensory modalities using SFA methods have shown to achieve good results [83].

8. Conclusion

We proposed an online-learning algorithm that enables a humanoid robotic agent, such as an iCub, to incrementally acquire skills in order of increasing learning difficulty, from its onboard high-dimensional camera inputs and low-level kinematic joint maps, driven purely by its intrinsic motivation. The method combines our recently introduced active modular *Slow Feature* learning algorithm, called Curious Dr. MISFA and the *options* framework. We formally defined the underlying learning problem and provided experimental results conducted using an iCub humanoid robot to topple, grasp and pick-place a cup. To our knowledge, this is the **first** method that demonstrates continual curiosity-based skill acquisition from high-dimensional video inputs in humanoid robots.

Acknowledgements

We thank Sohrob Kazerounian and Alan Lockett's assistance in revising some of this paper. This work was funded through SNF grant #138219 (Theory and Practice of Reinforcement Learning II) and through the 7th framework program of the EU under grant #270247 (NeuralDynamics project).

References

[1] Honda. Asimo robot: `http://world.honda.com/ASIMO`.

[2] Boston-Dynamics. Petman (protection ensemble test mannequin) humanoid military robot: `http://www.bostondynamics.com/robot_petman.html`.

[3] G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori. The icub humanoid robot: An open platform for research in embodied cognition. In *Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems*, PerMIS '08, pages 50–56, New York, NY, USA, 2008. ACM.

[4] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: a survey. *Journal of AI research*, 4:237–285, 1996.

[5] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. Cambridge, MA, MIT Press, 1998.

[6] R. W. White. Motivation reconsidered: the concept of competence. *Psychological review*, 66(5):297, 1959.

[7] G. R. Norman and H. G Schmidt. The psychological basis of problem-based learning: a review of the evidence. *Academic medicine*, 67(9):557–65, 1992.

[8] H. Abut, editor. *Vector Quantization*. IEEE Press, Piscataway, NJ, 1990.

[9] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, 1986.

[10] P. Comon. Independent component analysis, A new concept? *Signal Processing*, 36:287–314, 1994.

[11] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[12] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Berlin, 3rd edition, 2001.

[13] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, August 2002.

[14] L. Wiskott and T. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.

[15] J. Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

[16] J. Schmidhuber. Learning unambiguous reduced sequence descriptions. In J. E. Moody, S. J. Hanson, and R. P. Lippman, editors, *Advances in Neural Information Processing Systems 4 (NIPS 4)*, pages 291–298. Morgan Kaufmann, 1992.

[17] J. Schmidhuber. Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879, 1992.

[18] S. Lindstädt. Comparison of two unsupervised neural network models for redundancy reduction. In M. C. Mozer, P. Smolensky, D. S. Touretzky, J. L. Elman, and A. S. Weigend, editors, *Proc. of the 1993 Connectionist Models Summer School*, pages 308–315. Hillsdale, NJ: Erlbaum Associates, 1993.

[19] M. Klapper-Rybicka, N. N. Schraudolph, and J. Schmidhuber. Unsupervised learning in lstm recurrent neural networks. In *Lecture Notes on Comp. Sci. 2130, Proc. Intl. Conf. on Artificial Neural Networks (ICANN-2001)*, pages 684–691. Springer: Berlin, Heidelberg, 2001.

[20] O. C. Jenkins and M. J. Matarić. A spatio-temporal extension to isomap nonlinear dimension reduction. In *Proceedings of the twenty-first international conference on Machine learning*, page 56. ACM, 2004.

[21] H. Lee, Y. Largman, P. Pham, and A.Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems*, pages 1096–1104.

[22] S. Singh, A. G. Barto, and N. Chentanez. Intrinsically motivated reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1281–1288, 2004.

[23] Y. Girdhar, D. Whitney, and G. Dudek. Curiosity Based Exploration for Learning Terrain Models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[24] A. Stout and A. G Barto. Competence progress intrinsic motivation. In *Development and Learning (ICDL), 2010 IEEE 9th International Conference on*, pages 257–262. IEEE, 2010.

[25] L. Pape, C. M. Oddo, M. Controzzi, C. Cipriani, A. Förster, M. C. Carrozza, and J. Schmidhuber. Learning tactile skills through curious exploration. *Frontiers in neurorobotics*, 6, 2012.

[26] S. Hart, S. Sen, and R. A. Grupen. Intrinsically motivated hierarchical manipulation. In *Proceedings of the 2008 IEEE Conference on Robots and Automation (ICRA)*, pages 3814–3819, 2008.

[27] G. Konidaris, S. Kuindersma, R. Grupen, and A. G. Barto. Autonomous skill acquisition on a mobile manipulator. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 1468–1473, 2011.

[28] L. Gisslén, M. Luciw, V. Graziano, and J. Schmidhuber. Sequential constant size compressors for reinforcement learning. In *Artificial General Intelligence*, pages 31–40. Springer, 2011.

[29] M. B. Ring. *Continual Learning in Reinforcement Environments*. PhD thesis, University of Texas at Austin, 1994.

[30] J. Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187, 2006.

[31] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

[32] J. Schmidhuber. Curious model-building control systems. In *Proceedings of the International Joint Conference on Neural Networks, Singapore*, volume 2, pages 1458–1463. IEEE press, 1991.

[33] J. Storck, S. Hochreiter, and J. Schmidhuber. Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the International Conference on Artificial Neural Networks, Paris*, volume 2, pages 159–164. EC2 & Cie, 1995.

[34] J. Schmidhuber. Artificial curiosity based on discovering novel algorithmic predictability through coevolution. In *Congress on Evolutionary Computation (CEC)*, pages 1612–1618. IEEE Press, 1999.

[35] J. Schmidhuber. Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. *Connection Science*, 18(2):173–187, 2006.

[36] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990-2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230 –247, 2010.

[37] S. Lange and M. Riedmiller. Deep learning of visual control policies. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pages 265–270.

[38] M. Franzius, H. Sprekeler, and L. Wiskott. Slowness and sparseness lead to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, 3(8):e166, 2007.

[39] R. Legenstein, N. Wilbert, and L. Wiskott. Reinforcement learning on slow features of high-dimensional input streams. *PLoS Computational Biology*, 6(8), 2010.

[40] P. Földiák and M. P. Young. Sparse coding in the primate cortex. *The handbook of brain theory and neural networks*, 1:895–898, 1995.

[41] G. Mitchison. Removing time variation with the anti-hebbian differential synapse. *Neural Computation*, 3(3):312–320, 1991.

[42] G. Wallis and E.T. Rolls. Invariant face and object recognition in the visual system. *Progress in Neurobiology*, 51(2):167–194, 1997.

[43] L. Wiskott. Estimating driving forces of nonstationary time series with slow feature analysis. *arXiv preprint cond-mat/0312317*, 2003.

[44] H. Sprekeler, T. Zito, and L. Wiskott. An extension of slow feature analysis for nonlinear blind source separation. *Journal of Machine Learning Research*, 15:921–947, 2014.

[45] V. R. Kompella, M. Luciw, and J. Schmidhuber. Incremental slow feature analysis. In *Proc. 20th International Joint Conference of Artificial Intelligence (IJCAI)*, pages 1354–1359, 2011.

[46] V. R. Kompella, M. Luciw, and J. Schmidhuber. Incremental slow feature analysis: Adaptive low-complexity slow feature updating from high-dimensional input streams. *Neural Computation*, 24(11):2994–3024, 2012.

[47] M. Luciw*, V. R. Kompella*, S. Kazerounian, and J. Schmidhuber. An intrinsic value system for developing multiple invariant representations with incremental slowness learning. *Frontiers in Neurorobotics, *Joint First Authors*, 7, 2013.

[48] V. R. Kompella, M. Luciw, M. Stollenga, L. Pape, and J. Schmidhuber. Autonomous learning of abstractions using curiosity-driven modular incremental slow feature analysis. In *Proc. of the Joint Conference on Development and Learning and Epigenetic Robotics (ICDL-EPIROB)*, pages 1–8, San Diego, 2012. IEEE.

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

[50] M. Stollenga, L. Pape, M. Frank, J. Leitner, A. Förster, and J. Schmidhuber. Task-relevant roadmaps: a framework for humanoid motion planning. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5772–5778. IEEE, 2013.

[51] I. Menache, S. Mannor, and N. Shimkin. Q-cut–dynamic discovery of sub-goals in reinforcement learning. In *Machine Learning: ECML 2002*, pages 295–306. Springer, 2002.

[52] O. Şimşek and A. G. Barto. Skill characterization based on betweenness. In *NIPS'08*, pages 1497–1504, 2008.

[53] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *The Journal of Machine Learning Research*, 4:1107–1149, 2003.

[54] M. B. Ring. Child: A first step towards continual learning. *Machine Learning*, 28(1):77–104, 1997.

[55] B. Bakker and J. Schmidhuber. Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In F. Groen et al., editor, *Proc. 8th Conference on Intelligent Autonomous Systems IAS-8*, pages 438–445, Amsterdam, NL, 2004. IOS Press.

[56] S. W. Hart. *The development of hierarchical knowledge in robot systems*. PhD thesis, University of Massachusetts Amherst, 2009.

[57] M. Huber and R. A. Grupen. A hybrid discrete event dynamic systems approach to robot control. *Univ. Mass., Dept. Comput. Sci., Amherst, MA, Tech. Rep*, pages 96–43, 1996.

[58] G. Konidaris and A. G. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems*, pages 1015–1023, 2009.

[59] G. Konidaris, S. Kuindersma, A. G. Barto, and R. Grupen. Constructing skill trees for reinforcement learning agents from demonstration trajectories. In *Advances in Neural Information Processing Systems*, pages 1162–1170, 2010.

[60] J. Mugan and B. Kuipers. Autonomous learning of high-level states and actions in continuous environments. *IEEE Transactions on Autonomous Mental Development*, 4(1):70–86, 2012.

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

[62] H. Ngo, M. Luciw, A. Förster, and J. Schmidhuber. Learning skills from play: Artificial curiosity on a Katana robot arm. In *Proceedings of the International Joint Conference of Neural Networks (IJCNN)*, pages 1–8, June 2012.

[63] H. Ngo, M. Luciw, A. Förster, and J. Schmidhuber. Confidence-based progress-driven self-generated goals for skill acquisition in developmental robots. *Frontiers in Psychology*, 4, 2013.

[64] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

[65] N. Sprague. Predictive projections. In *Proceedings of the International Joint Conference of Artificial Intelligence (IJCAI)*, pages 1223–1229, 2009.

[66] H. Sprekeler. On the relation of slow feature analysis and laplacian eigenmaps. *Neural Computation*, 23(12):3287–3302, 2011.

[67] S. Mahadevan and M. Maggioni. Proto-value functions: A laplacian framework for learning representation and control in markov decision processes. *Journal of Machine Learning Research*, 8(2169-2231):16, 2007.

[68] M. Luciw and J. Schmidhuber. Low complexity proto-value function learning from sensory observations with incremental slow feature analysis. In *Proc. 22nd International Conference on Artificial Neural Networks (ICANN)*, pages 279–287, Lausanne, 2012. Springer.

[69] J. Schmidhuber. Learning to generate sub-goals for action sequences. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 967–972. Elsevier Science Publishers B.V., North-Holland, 1991.

[70] J. Schmidhuber and R. Wahnsiedler. Planning simple trajectories using neural subgoal generators. In J. A. Meyer, H. L. Roitblat, and S. W. Wilson, editors, *Proc. of the 2nd International Conference on Simulation of Adaptive Behavior*, pages 196–202. MIT Press, 1992.

[71] M. Wiering and J. Schmidhuber. HQ-learning. *Adaptive Behavior*, 6(2):219–246, 1998.

[72] V. R. Kompella, M. F. Stollenga, M. Luciw, and J. Schmidhuber. Explore to see, learn to perceive, get the actions for free: Skillability. In *International Joint Conference on Neural Networks (IJCNN)*, pages 2705–2712. IEEE, 2014.

[73] V. R. Kompella. *Slowness Learning for Curiosity-Driven Agents*. PhD thesis, Informatics Department, Università della Svizzera Italiana, 2014.

[74] A. Saltelli, K. Chan, E. M. Scott, et al. *Sensitivity analysis*, volume 134. Wiley New York, 2000.

[75] I. D. Guedalia, M. London, and M. Werman. An on-line agglomerative clustering method for nonstationary data. *Neural Computation*, 11(2):521–540, 1999.

[76] D. Zhang, D. Zhang, S. Chen, K. Tan, and K. Tan. Improving the robustness of online agglomerative clustering method based on kernel-induce distance measures. *Neural processing letters*, 21(1):45–51, 2005.

[77] E. W. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.

[78] Y. Zhang, H. Wu, and L. Cheng. Some new deformation formulas about variance and covariance. In *Modelling, Identification & Control (ICMIC), 2012 Proceedings of International Conference on*, pages 987–992. IEEE, 2012.

[79] D. Wierstra, T. Schaul, J. Peters, and J. Schmidhuber. Natural evolution strategies. In *IEEE World Congress on Computational Intelligence*, pages 3381–3387. IEEE, 2008.

[80] J. Weng, Y. Zhang, and W. Hwang. Candid covariance-free incremental principal component analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8):1034–1040, 2003.

[81] D. Peng, Z. Yi, and W. Luo. Convergence analysis of a simple minor component analysis algorithm. *Neural Networks*, 20(7):842–850, 2007.

[82] M. Luciw, V. R. Kompella, and J. Schmidhuber. Hierarchical incremental slow feature analysis. In *Workshop on Deep Hierarchies in Vision*, Vienna, 2012.

[83] S. Höfer, M. Spranger, and M. Hild. Posture recognition based on slow feature analysis. In *Language Grounding in Robots*, pages 111–130. Springer, 2012.