Autonomous Learning of Abstractions using Curiosity-Driven Modular Incremental Slow Feature Analysis

Varun Raj Kompella, Matthew Luciw, Marijn Stollenga, Leo Pape and Jürgen Schmidhuber
IDSIA, Galleria 2, Manno-Lugano 6928, Switzerland
Email: {varun,matthew,marijn,pape,juergen}@idsia.ch

***Abstract*—To autonomously learn behaviors in complex environments, vision-based agents need to develop useful sensory abstractions from high-dimensional video. We propose a modular, curiosity-driven learning system that autonomously learns multiple abstract representations. The policy to build the library of abstractions is adapted through reinforcement learning, and the corresponding abstractions are learned through incremental slow-feature analysis (IncSFA). IncSFA learns each abstraction based on how the inputs change over time, directly from unprocessed visual data. Modularity is induced via a gating system, which also prevents abstraction duplication. The system is driven by a curiosity signal that is based on the learnability of the inputs by the current adaptive module. After the learning completes, the result is multiple slow-feature modules serving as distinct behavior-specific abstractions. Experiments with a simulated iCub humanoid robot show how the proposed method effectively learns a set of abstractions from raw un-preprocessed video, to our knowledge the first curious learning agent to demonstrate this ability.**

I. Introduction

One of the challenging problems faced by an autonomous agent in a complex environment is how to derive useful behaviors in the presence of an abundance of sensory information. Typical high-dimensional input streams would need to be represented by a set of compact but discriminative *feature abstractions* mapping complex observation sequences to a low-dimensional space. These lower-dimensional representations will then enable a subsequent reinforcement learner to generate intelligent behaviors. Recently, an unsupervised learning technique called incremental slow feature analysis (IncSFA) [6] was introduced that learns useful sensory abstractions from high-dimensional video. Slow-features depend on how the inputs change over time [18] and can be used to learn behavior-specific abstractions [5], [11]. Learning distinct behaviors requires learning different abstractions. How can multiple abstractions be learned in the absence of a teacher? We propose here a novel curiosity-driven reinforcement learning framework called the Curiosity-Driven Modular Incremental Slow Feature Analysis (Curious Dr. MISFA), which facilitates autonomous learning of multiple slow-feature modules that serve as distinct behavior-specific abstractions.

The *Options* framework [17] formalized planning over temporally extended courses of actions (*temporal abstractions*) via the semi-Markov decision process. Each option is applicable over part of the world, has its own subgoal(s), and has its own policy. Each option has a set of initiation states (from which the option can be started), a policy for action selection, and a termination probability upon each state. Konidaris et al. in [7], [8] show how each option might be assigned with an abstraction from a library of many sensori-motor abstractions, potentially simplifying the learning problem. However, these abstractions have typically been hand-designed. In our model, we modify the option definition to include *feature-abstraction* as a part of it and explore ways to learn these feature-abstractions of several options using temporal-difference methods. RL on abstraction-based options have been applied to challenging domains such as those of humanoid robotics [9], in which learning was assisted by human-demonstration. These rely on teacher-given demonstrations, explicit task-descriptions by humans and externally defined goals for guidance. How a compact set of useful abstractions might be learned from exploration in such challenging domains remains an open problem. Curious Dr. MISFA will build a library of abstractions for such complex domains without external guidance.

In this paper, we focus purely on learning features from the environment without any external motivation. In this case, the agent needs to be self-motivated, i.e., curious. The *Formal Theory of Fun and Creativity* [15] mathematically formalizes driving forces behind all kinds of curious and creative behavior. A creative agent needs two learning components: a reinforcement learner and an adaptive encoder/predictor/compressor of the agent's growing history of perceptions and actions. The learning progress of the encoder becomes an intrinsic reward for the reinforcement learner [14].

The learning module in Curious Dr. MISFA has a output estimator, which is adaptively improving a model of *feature responses*. The agent is rewarded whenever the error in the model reduces. Once some level of accuracy is reached, the module is frozen and added to the abstraction library. In this way, without any prior knowledge of the environment, Curious Dr. MISFA curiously and incrementally builds modules encoding all the SFA-encodable regularities in the world. Once learned, the various slow-feature abstractions facilitate learning of several potentially intelligent behaviors.

II. Slow Feature Analysis

Our feature learning method is incremental SFA. Slow feature analysis [18] is an unsupervised learning technique

that extracts features from an input 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 $\mathbf{g}(x) = [g_1(\mathbf{x}), ..., g_J(\mathbf{x})]^T$, 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 $j \in \{1, ..., J\}$

$$\Delta_j := \Delta(y_j) := \langle \dot{y}_j^2 \rangle \quad \text{is minimal}- \tag{1}$$

under the constraints

$$\langle y_j \rangle = 0 \quad \text{(zero mean)}, \tag{2}$$

$$\langle y_j^2 \rangle = 1 \quad \text{(unit variance)}, \tag{3}$$

$$\forall i < j : \langle y_i y_j \rangle = 0 \quad \text{(decorrelation and order)}, \tag{4}$$

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 (4) ensures that different functions g_j do not code for the same features. The other constraints (2) and (3) avoid trivial constant output solutions.

Slow features are useful for RL. SFs approximate proto-value functions [12], [16] from sampled observations of a Markov Decision Process (MDP). They are approximations of the low-order eigenvectors of the graph Laplacian matrix representing the MDP. The approximate nature depends on the quality of the observation space. Theoretical analysis shows that just a few of these features can capture the global characteristics of some Markovian processes [2], [3].

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 solution complexity is cubic in the input dimension I. Incremental SFA (IncSFA) [6], however, has linear update complexity, and can adapt the features to new observations, achieving the slow-feature objective robustly in open-ended learning environments. Since Curious Dr. MISFA explores the world in an open-ended fashion, we use IncSFA instead of the batch method.

III. Curious Dr. MISFA

A. Architecture

Curious Dr. MISFA has a high-level architecture similar to the one introduced by Barto et al. [1]. The environment is divided into an *internal environment* and an *external environment* (Figure 1). The former contains an RL agent and an intrinsic-motivation block that generates internal rewards to the agent. The shaded box represents Curious Dr. MISFA. The external environment has a finite set of primitive states $\mathcal{S}$. At each time step the agent perceives the state $s \in \mathcal{S}$ by carrying out an action $a \in \mathcal{A}$ and makes an observation (potentially high-dimensional vector) $\mathbf{x} \in \mathbb{R}^I$.

The *internal environment* has a finite set of temporally-extended courses of actions $\mathcal{O} = \{O_1, O_2, ...O_n\}$, where

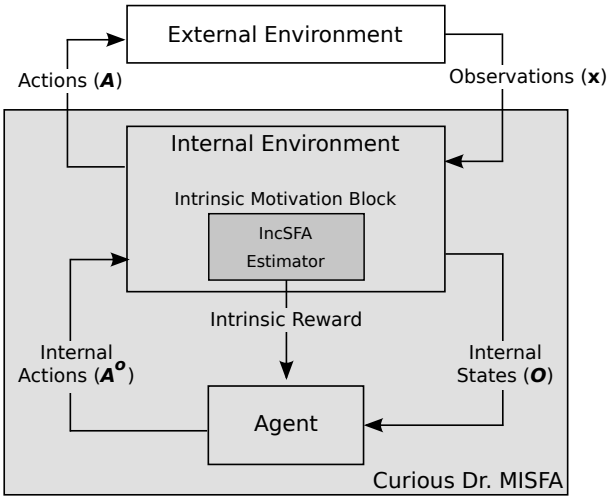

Fig. 1. Curious Dr. MISFA System Architecture. See text for details.

O_i is similar to an *option* [17]. Each option here is a five-tuple: $O = \langle \mathcal{I}, \beta, \pi^{\mathbf{x}}, \Phi, \pi^{\mathbf{y}} \rangle$. $\mathcal{I} : \mathcal{S} \rightarrow [0, 1]$ is the initiation set indicator function, which is 1 in states where the option applies and 0 elsewhere. β is the option termination condition (which will determine when and where the option ends). $\pi^{\mathbf{x}} : \mathcal{S}^+ \times \mathcal{A} \rightarrow [0, 1]$ is a hard-coded *exploration* policy, such as a random walk within the state space where the option is applicable ($\mathcal{S}^+ \subseteq \mathcal{S}$). The exploration policy is used to collect input observations ($\mathbf{x}$). Φ is the option's abstraction set, initially unknown, these are learned features which map the I-dimensional observation vector to a J-dimensional encoded observation: $\Phi : \mathbf{x} \mapsto \mathbf{y}$, where $J << I$. And finally, $\pi^{\mathbf{y}} : \mathcal{S}^+ \times \mathcal{A} \rightarrow [0, 1]$ is an *exploitation* policy, meant for improving via RL while using the abstraction output $\mathbf{y}$ as an input. We will not deal with improving the exploitation policy in this paper, but we wish to lay the groundwork for future work, which will do so.

When we say an option O_i *executes*, we mean it starts in the current state $s \in \mathcal{I}_i$ and its exploration policy $\pi_i^{\mathbf{x}}$ takes control until it hits a termination condition β_i. However, a termination condition can only occur in states where at least one option is applicable.

The goal of Curious Dr. MISFA is to build the *abstraction library*,

$$\mathbf{\Phi}^{\mathcal{L}} = \{\Phi_1^{\mathcal{L}}, \Phi_2^{\mathcal{L}}, ..., \Phi_{\leq n}^{\mathcal{L}}\}, \tag{5}$$

where each $\Phi_i^{\mathcal{L}}$ refers to a learned abstraction in the order that the abstraction from an "easily learnable" option is learned first.

Unlike the *options* framework presented in [17], we refer to each option as an *internal-state* ($O_i \in \mathcal{O}$) that the agent observes in the internal environment (see Figure 1). When the agent shifts to one of the internal-states, it executes the corresponding option. Note that when an option O_i is executing, this implies that the agent's current internal-state is O_i.

When the execution of the current option terminates, the agent can take two *internal-actions*, $\mathcal{A}^o = \{\text{stay, switch}\}$.

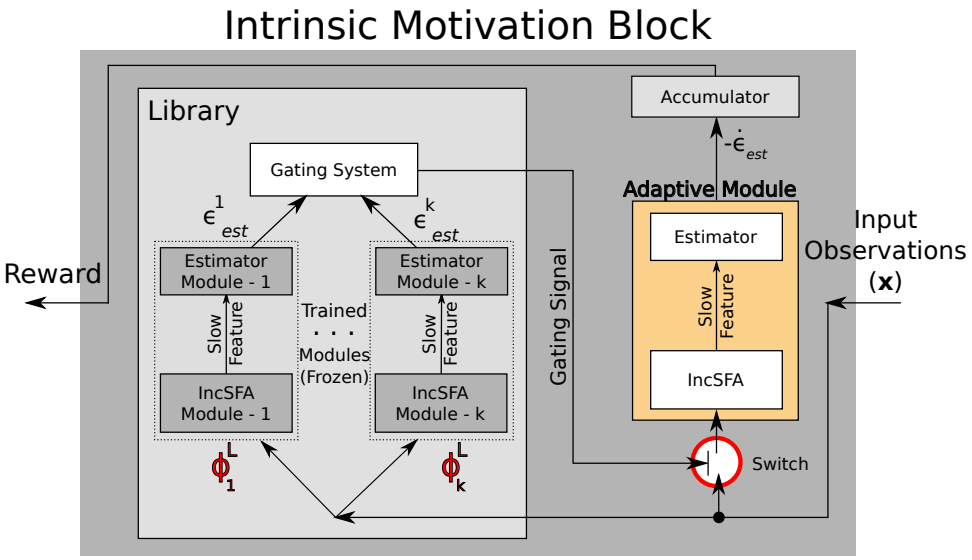

Fig. 3. Inner details of the Intrinsic Motivation Block.

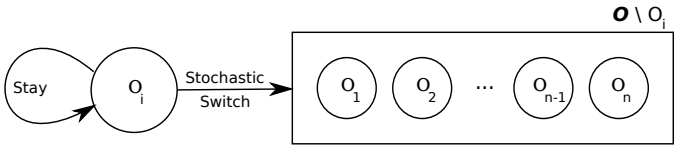

Fig. 2. The agent has an internal-state space including all n options, and two actions: stay (in the current option) or switch stochastically to another. It selects an internal-action (stay or switch) whenever an option terminates.

The internal-action *stay* makes the agent to stay in the same internal-state and executes the corresponding option again. While *switch* randomly (uniform) shifts the agent's internal state to one of the other internal-states (stochastic switching)[1].

Figure 2 shows the state-diagram of the internal environment. The objective of the RL agent is to find a policy: $\mu : \mathcal{O} \times \mathcal{A}^o \rightarrow [0,1]$ that selects which option to execute over time to build the abstraction library $\mathbf{\Phi}^{\mathcal{L}}$.

At any time the input stream $\mathbf{x}(t), \mathbf{x}(t+1), ...$ is a function of one of the n input stream-generating options $O_1, .., O_n$. Whichever option is currently executing runs its exploration policy until it terminates, and this exploration policy generates a stream of observation vectors. While some options may lead to random incompressible observation sequences, others observation streams may contain learnable regularities.

Figure 3 shows the inner details of the intrinsic motivation block. It has an *adaptive module* and a library of k (equal to 0 at t=0) learned *frozen modules*. Each module is a combination of an IncSFA module and its estimated output model (estimator).

To learn each abstraction, a single training module (see Figure 3) continuously updates on the stream of observations. The decrease of estimation error of the learning module $-\dot{\epsilon}_{est}$ is an intrinsic reward which is accumulated until the current option terminates. Upon termination, the cumulative reward is then given to the agent, which reinforces its future internal-actions.

[1]Note that the goal of Curious Dr. MISFA is to learn to build abstractions reflecting the given input streams, not from combining them, which would lead to a combinatorial explosion of observation sequences. Stochastic switching becomes important to prevent IncSFA from picking up regularities generated by a deterministic source-switching policy (n-armed bandit).

B. Abstraction Updating

IncSFA: There is always only a single adaptive IncSFA (see Figure 3) continually updated on each sample. To summarize briefly (details: [6]):

The adaptive module's IncSFA incrementally updates existing slow-feature estimates after each derivative measurement: $\dot{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{x}(t-1)$, where $\mathbf{x}(t)$ is the high-dimensional input observation (such as an image) at time t and the derivative is computed via backward-difference approximation.

The update for each slow-feature vector $\mathbf{w}_i$ from 1 to J, where J is the number of slow-features used is

$$\mathbf{w}_i \leftarrow (1-\eta^{SFA})\mathbf{w}_i - \eta^{SFA}\left((\dot{\mathbf{x}}\cdot\mathbf{w}_i)\,\dot{\mathbf{x}} + \gamma \sum_{j}^{i-1}(\mathbf{w}_j\cdot\mathbf{w}_i)\mathbf{w}_j\right) \quad (6)$$

where η_{SFA} is a learning rate. This update is based on anti-Hebbian learning with an additional Gram-Schmidt term inside the summation that enforces different features to be orthogonal. $\mathbf{w}_j$ are the "lower-order" slow-features, and γ is larger than the first eigenvalue of $\dot{\mathbf{x}}$. We compute the most significant PC of $\dot{\mathbf{x}}$ with an incremental PCA technique to get this scalar. After updating, a feature is normalized for stability. The feature output does *not* use the derivative signal, but is an instantaneous function:

$$\mathbf{y}(\mathbf{t}) = \mathbf{x}(\mathbf{t})^T\mathbf{w}(t) \quad (7)$$

In practice, we found it beneficial for images to first pass $\mathbf{x}$ through a simultaneously learning spatial image compressor, such as an neural net autoencoder [5]. (see Section IV-B for more details)

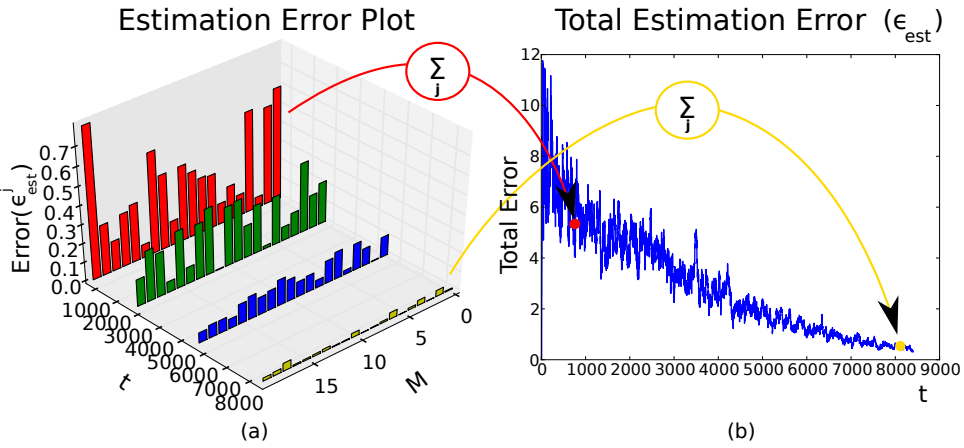

Fig. 4. An example progress over time of the estimator. (a) The change in estimation error over time is shown. Here we use 20 instances of ROC-nodes to estimate the input. Over the time the errors go down. (b) Here the sum of estimation-errors is shown. The total error goes down as the estimations improve. Note that this decrease is the reward used by the agent to choose between the 'stay' and 'switch' actions.

Estimator: The agent's adaptive module additionally has an estimator that maintains estimates of the current slow feature outputs. If the feature outputs become stable, the error of the estimator decreases. This decrease motivates the agent to focus on the options where it makes the most progress.

The slow feature outputs can change quite rapidly during the training phase. The estimator therefore has to be able to change its estimates to this initially non-stationary input, but still build up a good robust estimate when the input becomes stable. To this end, we use a kernelized agglomerative clustering algorithm called the Robust Online Clustering algorithm (ROC) [19], [4]. The method is similar to a kernelized k-means algorithm, but by using a merging approach, ROC can quickly adjust to non-stationary input distributions by directly adding a new cluster for the newest input sample. k-means in contrast would slowly have to move an existing cluster to a new point. We further enhanced its performance to non-stationary distributions by adding a forgetting parameter.

The estimator uses M instances (nodes) of the ROC algorithm, each containing and updating their own clusters. Each node is assigned to a part of the observation space by a binning function, mapping the observations to an integer $b : \mathcal{S} \rightarrow w,\ 0 \leq w < M, w \in \mathbb{N}$. Then the current slow feature output $\mathbf{y}$ becomes represented by the ROC instance corresponding to w.

C. Estimation error and Intrinsic Reward

Each estimator node j has an associated error ϵ^j_{est}. These errors are initialized to 0 and then updated whenever the node is activated by:

$$\epsilon^j_{est}(t) \quad = \quad \min_w ||\mathbf{y}(t) - \mathbf{v}_w|| \tag{8}$$

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:

$$\epsilon_{est}(t) \quad = \quad \sum_{j=1}^{M} \epsilon^j_{est}(t) \tag{9}$$

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 ($\dot{\epsilon}_{est}$) as follows:

$$R_a^{o,o'} := (1-\eta)\ R_a^{o,o'} + \eta \sum_{t}^{t+T} -\dot{\epsilon}_{est}(t) \tag{10}$$

where $0 < \eta < 1$ is a discount factor, T is the duration of the current option until its termination, $(o, o') \in \{O_1, ..., O_n\}$ and $a \in \{\text{stay, switch}\}$.

Figure 4 shows an example progress of an estimator (with 20 ROC nodes mapped to say 20 states) over time. Figure 4(a) shows the estimation error of each node that is mapped to a part of the observation space. As the adaptive IncSFA features stabilize, the estimation error at each node decreases over time. Figure 4(b) shows the plot of total estimation error with time. The decrease in this error is used for computing the intrinsic reward.

D. RL Policy Generation

The transition-probability model P of the internal environment is similar to a complete graph and is given by:

$$P^{stay}_{ij} \quad = \quad \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases} \tag{11}$$

$$P^{switch}_{ij} \quad = \quad \begin{cases} 0, & \text{if } i = j \\ \frac{1}{N-1}, & \text{if } i \neq j \end{cases} \tag{12}$$

$\forall i, j \in [1, ..., N]$. Using the current updated model of the reward function R and the internal-state transition-probability model P, we use model-based Least Squares Policy Iteration [10] to generate the agent's policy ($\mu : \mathcal{O} \times \mathcal{A}^o \rightarrow [0, 1]$) for the next iteration. The agent uses decaying ϵ-greedy strategy over the agent's policy to carry out an internal-action (stay or switch) for the next iteration.

E. Module Freezing and New Module Creation

Once the adaptive (training) module's estimation error gets lower than a threshold, the agent freezes and saves the IncSFA/estimator module, resets the ϵ-greedy value and creates/starts training a new module.

F. Abstraction Assignment

The already trained (frozen) modules represent our learned library of abstractions $\mathbf{\Phi}^{\mathcal{L}}$ (Eq. 5). If a *trained* module's estimation error within an option is below a threshold, that option is assigned that module's abstraction and the adaptive training module will be prevented from learning via a "gating signal" (see Figure 3). There will no intrinsic reward in this case. Hence the training module will encode only data from input streams that were not encoded earlier. Input badly encoded by all other trained modules serve to train the adaptive module.

IV. Experiments and Results

A. Oscillatory Audio Signals

To illuminate the system's learning behavior over time, we conduct three tests in a toy environment. There are 500 states in the external environment ($1 \leq s \leq 500$) and either three or four options (internal states—e.g., $\mathcal{O} \in \{O_1, O_2, O_3\}$). Each option's exploration policy simply advances the state to the next (and from state 500 to state 1). Termination probabilities are 1 at every 50^{th} state, and 0.0 elsewhere, hence the agent has to decide on an internal-action (stay or switch) every 50 time steps.

Test 1: Three Encodable Signals. In the first test, the audio streams generated by each option O are encodable by IncSFA.

$$\text{if } O = O_1 \quad : \quad \begin{cases} x_1(t) = \sin(t) + \cos(11\ t)^2 \\ x_2(t) = \cos(11\ t) \end{cases} \tag{13}$$

$$\text{if } O = O_2 \quad : \quad \begin{cases} x_1(t) = \sin(11\ t) \\ x_2(t) = \cos(11\ t)^2 + \sin(2\ t) \end{cases} \tag{14}$$

$$\text{if } O = O_3 \quad : \quad \begin{cases} x_1(t) = \sin(11\ t) \\ x_2(t) = \cos(11\ t)^2 + \sin(0.5\ t) \end{cases} \tag{15}$$

where t is s mapped to $[0, 2\pi]$. All signals contain both quickly and slowly-varying components. The slowest feature in the signal $O_1(t)$ is $y(t) = x_1(t) - x_2(t)^2 = \sin(t)$ [18]. Similarly, the slowest features in the signals $O_2(t)$ and $O_3(t)$ are $y(t) = x_1(t)^2 + x_2(t) = 1 + \sin(2\ t)$ and $y(t) = x_1(t)^2 + x_2(t) = 1 + \sin(0.5\ t)$, respectively.

Test 2: Two Encodable and one Random Signals. In the second experiment, we have two stationary signals from the first experiment and one random signal (uniform noise).

Test 3: One Encodable and Three Random Signals. Three options produce random signals, only one generates a signal with learnable regularity.

Experiments. In all cases Curious Dr. MISFA starts with a single adaptive abstraction module (*i.e.* k=0 frozen modules) and random RL policy μ (see Sec. III-D). When selected each option execute its exploration policy. The agent explores using an epsilon-greedy, on-policy strategy until it stops receiving intrinsic rewards, which indicates that the abstraction library

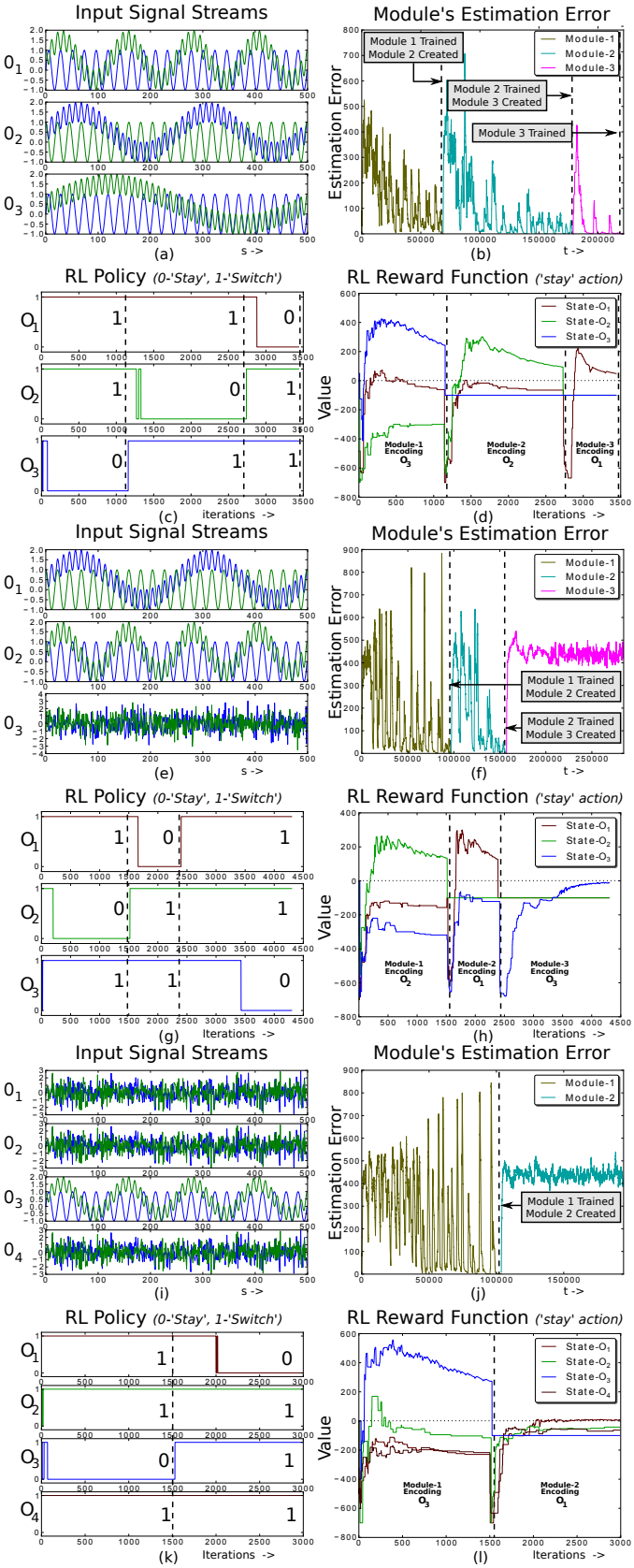

Fig. 5. **(a)-(d) Test 1 Result. (e)-(h) Test 2 Result. (i)-(l) Test 3 Result** (Figures are best viewed in color)

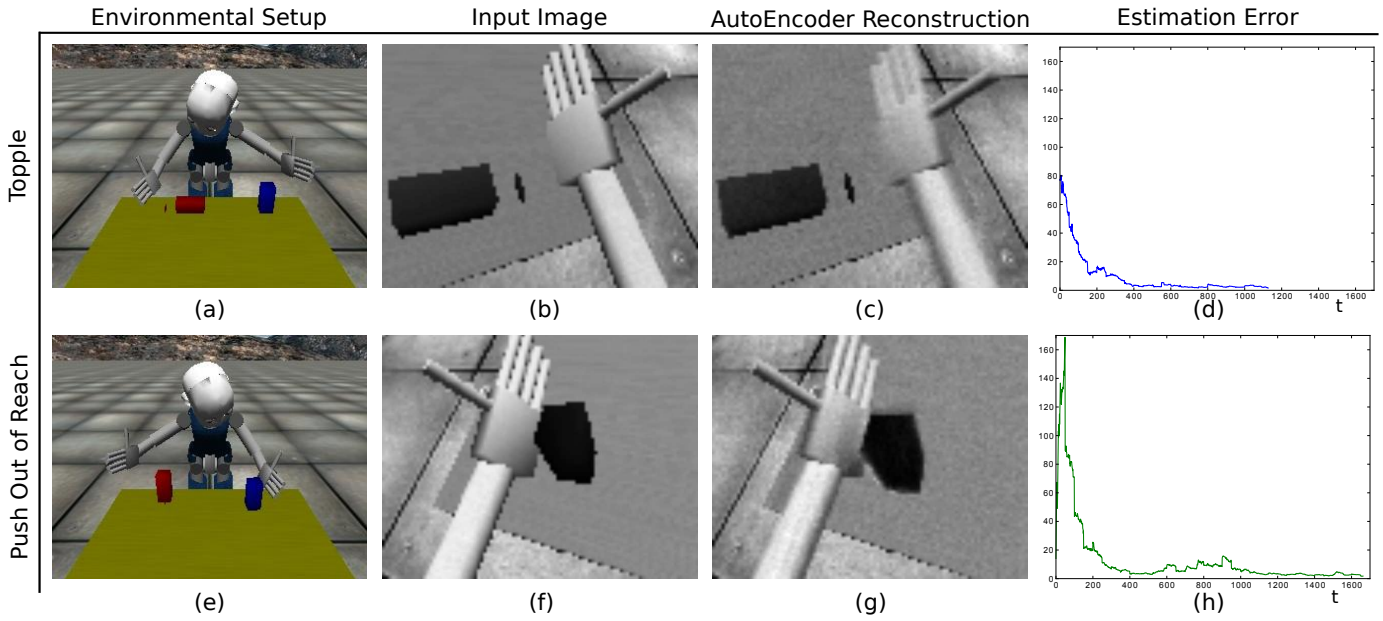

Fig. 6. **iCub Experiment:Topple** (a) Experimental setup: iCub observing its right arm's movements. (b) Sample input image. (c) Autoencoder reconstruction. (d) Estimation progress over time. **Push** (e) Experimental setup: iCub observing its left arm's movements. (f) Sample input image. (g) Autoencoder reconstruction. (h) Estimation progress over time. IncSFA takes more time to learn this regularity compared to toppling the object. (Figures are best viewed in color).

is complete.

Results are shown in Figs. 5. Refer parts (a)-(d) for Test 1 results. Part (a) shows signals generated in each option. Part (b) shows the evolution of estimation progress for each module. Since signal complexities are similar, the agent has no preference for any signal source, and the modules are learned in an arbitrary order. We see that when the estimation error of module-1 goes close to zero, the module is frozen and a new module-2 is now created. Part (c) shows the RL policy (μ) over algorithm iterations. Values $(0, 1)$ in the plot correspond to the actions (*stay*, *switch*) respectively. The policy (μ) converges very quickly and is stable until the corresponding adaptive module has been trained. It then automatically shifts to a new policy, where the observations are made from the un-encoded input streams. Part (d) plots the modeled reward function R of the RL agent for the stay action. During training, the reward of a state with a policy equal to 'stay', is higher, allowing the agent to train the estimator until it reaches a low estimation error. The ϵ-greedy exploration (see Sec. III-D) over the current policy allows the estimator to make continual progress, preventing the reward from dropping quickly before the estimation error reaches a low value. This is also evident from part (b), where the estimation error is not smooth and the spikes are a result of switching to other options. The upward slope of the spike (negative reward) is reflected in R for the option the agent is switched to. And the positive reward, due to the downward slope, is added to R when the agent switches back to the earlier option where it was making progress.

Refer to parts (e)-(h) of Figure 5 for Test 2 results. Part (e) shows signals generated in each option, where O_3 generates a random signal. Part (f) plots estimation error of created modules over time: The agent first learns modules for the stationary signals because here its IncSFA/estimator can make progress most quickly. This is also reflected by the Learned RL policy (part (g)) and the higher reward for modules 1 and 2 in part (h). Only after signals with quickly learnable regularities have been learned, the system tries to learn the random signal stream, but the corresponding module does not make much progress beyond learning the signal mean.

Refer to parts (i)-(l) of Figure 5 for Test 3 results. This result shows the agent's ability to pick out a learnable signal from a larger number of random signal input streams as shown in Part (i). Part (j) shows that the system generates only two modules, one for the regular signal, one for the noisy one. Initially, the agent makes similar progress with both the internal actions (switch or stay) for the noisy signals. This increases the probability for the agent to switch to the option that produces the regular signal. There the agent learns to select the stay action until the estimator has learned to estimate the IncSFA outcome as seen in parts (k) and (l).

Tests 2 and 3 also demonstrate that the system focuses on those parts of the environment where there are *learnable* regularities, as opposed to parts of the environment where its estimation error does not decrease.

B. Raw Video Streams in the iCub simulator

Environment. Here we present the results of an experiment conducted in the iCub dynamics (ODE-based) simulator [13]. The robot is placed at a table with two objects (Figure 6(a)), each in reach of one of its two arms. The red object (left) sits on a small nub on the table; the blue object (right) is placed directly on the table.

There are two options O_1 and O_2 and therefore two corresponding internal states. In O_1, the iCub randomly moves its left arm along the shoulder joint. When the iCub touches the red object, the object topples and falls sideways onto the table,

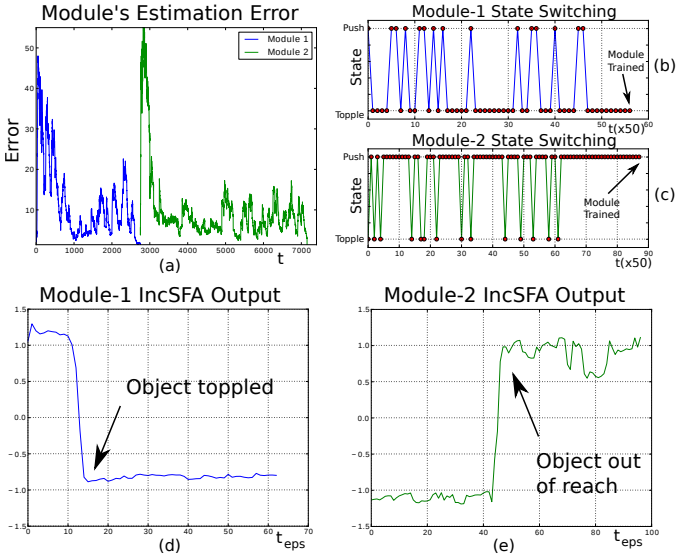

Fig. 7. (a) Learning progress of the modules over time. Module-1 learns to encode the 'toppled' abstraction and Module-2 encodes the 'pushed-out-of-reach' abstraction. (b)-(c) Convergence plots of the two modules. (Figures are best viewed in color)

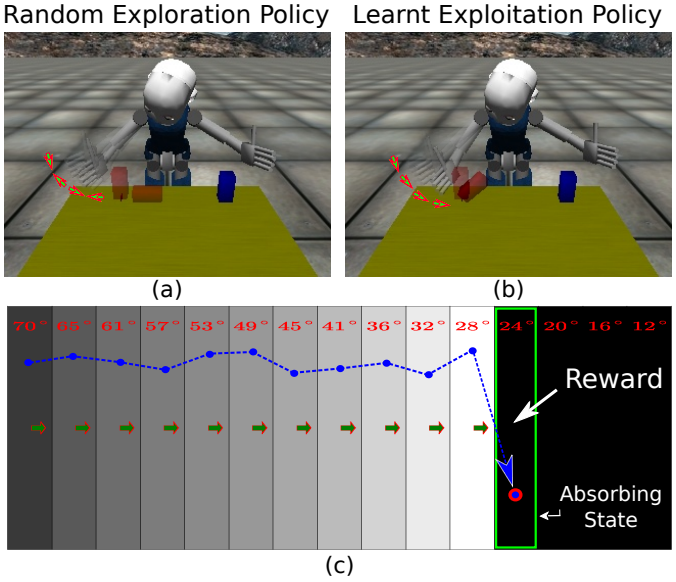

Fig. 8. (a) Random Exploration Policy used for abstraction learning. (b) An exploitation policy Learned using the frozen abstraction for an RL task to topple the object. (c) The optimal state action-value function for the task where the agent receives an internal reward for the step-transition of the abstraction output (blue dotted line) for each observation (joint angle). Transition happens as a consequence of the object being toppled. (Figures are best viewed in color).

and the option terminates. In O_2, the iCub randomly moves its right arm along the shoulder joint. When it touches the blue object, the object slides over the table. The iCub can push the blue object out of reach, in which case the option terminates. After termination, the robot's arm and the object is scripted to reset to an initial position. In both cases, the joint angles are discretized into 30 external states along the range of possible angles.

In either option, the robot moves its head to keep the object it is interacting with within its field of view. Raw high-dimensional monochrome video streams from the iCub's simulated cameras, downsampled to 75×100 pixels, form the sensory inputs to the algorithm. Figure 6(b) shows a sample input image. We used two autoencoders similar to the ones discussed in [5] for both eyes to compress the visual stream into a lower-dimensional input stream—both autoencoders have a single hidden layer with 100 hidden states. They are trained incrementally using standard error back-propagation. However, any other dimensionality reduction could be used, e.g., incremental PCA.

Results. Curious Dr. MISFA initially explores both options. If these options were to be trained individually, the result would be a step function that denotes whether the object is toppled or not [5] (for the right arm) or whether the object is pushed out of reach or not (for the left arm). The slope of the estimation error plots (see Figures 6(d) and (h)) of the slow-feature outputs for the toppling event is lower than the one for the push event, indicating that the topple event is easier to encode. After several episodes, the first module converges: object toppling is encoded via a step function. Once this module is frozen, the corresponding region generates no more internal rewards. The system now focuses on learning to encode object pushing. Once both the events are Learned, the robot has encoded all typical perceptive sequences, waiting for additional novel changes in the environment[2].

The robot autonomously learns two abstractions ((1) object toppled, (2) object pushed), and knows which abstraction applies in which option. This could potentially assist the learning of more complex behaviors.

V. Discussion

In the experiments presented above, Curious Dr. MISFA autonomously builds abstractions that encode the agent's interactions with the environment from raw pixels in an unsupervised manner. Once an abstraction is built, it can be used to learn several exploitation policies ($\pi^{\mathbf{y}}$) representing behaviors (or skills). Here is a basic example of learning an exploitation policy for the toppling-event discussed in Sec. IV-B. Figure 8(a) shows the random exploration policy ($\pi^{\mathbf{x}}$) used by the iCub for learning the abstraction. Figure 8(b) shows an exploitation policy Learned by using the frozen abstraction in an episodic-RL task through Q-learning. The iCub tries to predict the step transition in the Learned abstraction and receives internal rewards in doing so. Figure 8(c) shows state value, optimal action (left or right), and the abstraction output (blue dots) for each observation (15 discrete joint angles). Task-specific exploitation policies could be Learned more efficiently by using multi-spectral and manifold-learning techniques [12]. We leave this for future work.

Curious Dr. MISFA can prevent abstraction duplication and therefore permit generalization. It does not need to know the number of useful abstractions in advance. New ones are created once existing ones are insufficient to encode/predict input signals well.

[2]A video of the experiment can be found at http://www.idsia.ch/~kompella/videos/curious_dr_misfa.mp4

TABLE I
PARAMETERS THAT AFFECT THE WORKING OF CURIOUS DR. MISFA

Curious Dr. MISFA Paramters		
Learner	Parameters	Significance
IncSFA/AutoIncSFA	Learning Rate (η^{SFA})	A constant set for each experiment. Determines the performance of IncSFA. See [6] for more details.
ROC Estimator	No. of ROC Nodes (M)	Equal to the total number of observable primitive states in $\mathcal{S}$. Any other value also works but leads to appropriate quantization of the observed state space.
	Amnesic Parameter (l)	A constant set to 0.2 in the experiments. It determines the adaptiveness of the estimator.
RL (LSPI)	ϵ-greedy (ϵ)	Exploration/exploitation trade-off parameter. Initialized to 0.4 in our experiments.
	Decay constant (λ)	Decay term for ϵ-greedy ($0< \lambda < 1$). ϵ decays every time-step. Set to 0.995 in our experiments.
	Option termination condition (β)	Determines the option execution time. This parameter mainly affects the total experiment time. However, a very large option execution time could result in learning an abstraction before the option terminates. This will disrupt the order in which the abstractions are Learned.

We expect our system to scale to environments with numerous learnable regularities. In ongoing work we are testing it in increasingly complex environments using tree-based search involving the iCub simulation and the actual iCub robot.

Finally, Table I summarizes the main parameters to be tuned and their significance on the functioning of Curious Dr. MISFA.

VI. CONCLUSIONS

Curious Dr. MISFA is an autonomous, modular, incremental slow-feature-based reinforcement learner driven by learning progress. From raw camera inputs it automatically creates a library of abstractions, driven by an internally-generated curiosity signal that forces the system to focus on quickly *learnable* parts of its environment (instead of just focusing on parts where prediction error is high). In this sense, the system continually searches for the easiest-to-learn but not yet solvable task, ignoring parts of the environment that are too difficult to learn.

ACKNOWLEDGMENTS

We thank Mark Ring for his valuable comments on the paper. This work was funded through the 7th framework program of the EU under grants #231722 (IM-Clever project) and #270247 (NeuralDynamics project), and by Swiss National Science Foundation grant CRSIKO-122697 (Sinergia project).

REFERENCES

[1] A.G. Barto and O. Simsek. Intrinsic motivation for reinforcement learning systems. In *Proceedings of the Thirteenth Yale Workshop on Adaptive and Learning Systems*, pages 113–118, 2005.

[2] R.R. Coifman, S. Lafon, A.B. Lee, M. Maggioni, B. Nadler, F. Warner, and S.W. Zucker. Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *Proceedings of the National Academy of Sciences of the United States of America*, 102(21):7426, 2005.

[3] R.R. Coifman and M. Maggioni. Diffusion wavelets. *Applied and Computational Harmonic Analysis*, 21(1):53–94, 2006.

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

[5] V. R. Kompella, L. Pape, J. Masci, M. Frank, and J. Schmidhuber. Autoincsfa and vision-based developmental learning for humanoid robots. In *IEEE-RAS International Conference on Humanoid Robots*, Bled, Slovenia, 2011.

[6] 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.

[7] G. Konidaris and A. Barto. Efficient skill learning using abstraction selection. In *Proceedings of the Twenty First International Joint Conference on Artificial Intelligence*, pages 1107–1112, 2009.

[8] G. Konidaris and A.G. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in Neural Information Processing Systems*, 22:1015–1023, 2009.

[9] G. Konidaris, S. Kuindersma, A. Barto, and R. Grupen. Constructing skill trees for reinforcement learning agents from demonstration trajectories. *Advances in Neural Information Processing Systems*, 23:1162–1170, 2010.

[10] M.G. Lagoudakis and R. Parr. Model-free least-squares policy iteration. *Advances in neural information processing systems*, 2:1547–1554, 2002.

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

[12] S. Mahadevan. Proto-value functions: Developmental reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 553–560. ACM, 2005.

[13] Giorgio Metta, Giulio Sandini, David Vernon, Lorenzo Natale, and Francesco 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.

[14] 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.

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

[16] H. Sprekeler. On the relation of slow feature analysis and laplacian eigenmaps. *Neural computation*, pages 1–16, 2011.

[17] 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.

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

[19] D. Zhang, S. Chen, 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.