SEVENTH FRAMEWORK PROGRAMME

Project Acronym:	Xperience
Project Type:	IP
Project Title:	Robots Bootstrapped through Learning from Experience
Contract Number:	270273
Starting Date:	01-01-2011
Ending Date:	31-12-2015

XPERIENCE.ORG

Deliverable Number:	D2.2.2
Deliverable Title:	Motor Actions with focus on learning from examples encoded in action graphs
Type (Internal, Restricted, Public):	PU
Authors:	Tamim Asfour, Aleš Ude, Florentin Wörgötter, Miha Deniša, Mirko Waechter, and Rüdiger Dillmann
Contributing Partners:	KIT, JSI, UGOE

Contractual Date of Delivery to the EC:	31-01-2014
Actual Date of Delivery to the EC:	08-02-2014

Contents

Chapter 1

Summary

1.1 General Objective of WP2.2: Motor Actions

From the proposal: WP2.2 is primarily concerned with how to learn and obtain complex action sequences and how to organize and structure the acquired data to 1) generate advanced motor behaviours, e.g. by blending and sequencing of behaviours in the available data sets, and 2) interact with higher-level cognitive processes that mainly use discrete representations, thus providing the bridge for planning to access representations at the sensorimotor level and vice versa.

Methods like imitation learning, reinforcement learning and other exploratory approaches, which have been shown to be successful at the acquisition of motor knowledge, will be considered for implementation.

1.2 Summary of Results

In deliverable **D2.2.1** of Year 2 we investigated the structuring of the database of example trajectories. More specifically, we studied how to search in a hierarchical database of example movements to discover new action primitives [2] and how to generate cooperative behaviors using a database of cooperative trajectories [4]. Based on this previous work, deliverable **D2.2.2** addresses the following aspects

- the problem of generating new compliant reaching movements by searching a hierarchical database of example trajectories [DPAU13], and
- action sequence reproduction, where a higher-level representation, i. e. OACs, is used in the library [WSA$^+$13] instead of trajectories.

In addition, we included two papers that deal with basic research aspects concerning learning methodology. Here we have focused on a scientifically rather novel aspect of investigating the *combination* of different learning methods using network technologies. Specifically we have looked into

- a combination of reservoir computing, which is a supervised learning method for implementation of dynamic memories in networks, with reward modulated Hebbian learning, which allows for the learning of goal directed behaviors, and
- a combination of reinforcement learning with correlation based learning to reduce the problem of the "curse of dimensionality" in robotic reinforcement learning by speeding up initial convergence through the correlation based terms.

In the following we provide a short summary of this research. More details are available in the attached papers [DPAU13, WSA$^+$13, DWMM13, MKWM13].

1.3 Links to Other Workpackages

Deliverable **D2.2.2** and the earlier deliverable **D2.2.1** address theoretical issues in motor learning and thereby provide the basis for research in other workpackages, especially WP3.1, Structural Bootstrapping on sensorimotor experience, WP4.1, Cooperative Tasks, and WP5, System Integration and Demonstration. For example, the work on dynamic movement primitives and action graphs was used as a basis for structural bootstrapping at sensorimotor level in WP3.1 (see also **D3.1.1** and **D3.1.2**) and for the investigation and implementation of cooperative behaviors in WP4.1 (see also **D4.1.1** and **D4.1.2**).

Chapter 2

Description of Results

2.1 Synthesizing Compliant Reaching Movements by Searching a Database of Example Trajectories

In this work we address the problem of generating new compliant reaching movements by searching a structured database of example trajectories. The proposed learning framework is a multi-step process, where in the first step a human tutor teaches the robot how to perform a set of example reaching movements. In the second step, the recorded motion trajectories are executed with different velocities using a high gain feedback controller, for the purpose of learning corresponding torque control signals. The commanded torques are measured and stored together with the trajectory data, which is similar to research described in [4], where trajectories of a human performing the desired task were supplemented with trajectories of another person cooperating with him. We organized the recorded data in a hierarchical, graph-like structure, thereby providing the basis to search for new compliant trajectories, which can consist of parts of previously acquired example movements. The proposed approach can construct a complete representation of newly discovered movements, including the feedforward torque commands. Finally, in the last step, the motion is executed using a low gain feedback controller and the associated feedforward torque signal. This ensures sufficient tracking accuracy and at the same time compliant behavior, which allows smooth interaction with the environment and is safe for cooperative task execution with humans. The usefulness of the proposed method was shown on Kuka LWR robot. See also the attached paper [DPAU13].

2.2 Action Sequence Reproduction based on Automatic Segmentation and Object-Action Complexes

In this work we address the problem of automatic segmentation of human actions based on a library of Object-Action-Complexes (OACs) to provide a robot with the capability to recognize actions, adapt and reproduce them in new situations based on previously acquired action knowledge. In [WSA$^+$13], we address the problem by combining the formalism of Object-Action-Complexes ([3]) and Semantic Event Chains (SEC,[1].

The human demonstrations including a number of involved objects are captured by the optical tracking system (VICON), and segmented into semantic conclusive sub-actions by detecting relation changes between the objects and the human hand following the SEC concept. However, instead of using a 3D vision system, the demonstration was captured by the VICON system and thus the detection of object-hand relations was not based on color segmentation but on 3D Euclidian distances of the markers attached to the objects and the human. To detect object-hand relations, markers were grouped by the object they are attached to. Further, markers and marker groups are labeled for further post-processing. Based on the change of marker distances, key frames are detected when relations (touching, non-touching) between objects change. The key frames are used to divide the demonstration into action segments. The relations represent the world state and are later used for matching between action segments extracted from the demonstration and actions in the OAC library.

The OAC library was built in advance to store basic object manipulation and interaction skills enriched with preconditions and effects which are represented as the world state before and after the execution of an OAC. For the identification of action segments, a search in the OAC library is performed to find a matching OAC by comparing the preconditions and effects of all OACs in the library with the pre- and post-world state of a segmented action.

Finally, an observed sequence of OACs is parametrized for the execution on a humanoid robot (here: ARMAR-III) while adapting to change of the world state during execution. The feasibility of this approach is shown in an exemplary kitchen scenario, where the robot has to prepare a dough.

The details of the work are described in [WSA+13].

2.3 Basic Learning Theory Aspects

In this work we address the problem of the "curse of dimensionality" in robotics's reinforcement learning (RL). In general, RL can derive a policy for motor control according to the (delayed) reward signal. However, for high-dimensional continuous-state systems, using it without any prior knowledge (predefined control parameters), environment or system models, or appropriate guidance often requires long learning times. According to this, we developed a biologically-inspired learning model that combines correlation-based learning using input correlation learning (ICO learning) and reward-based learning within continuous actor-critic RL, thereby working as a dual learner system. This combinatorial learning framework suggests how a prior knowledge can be provided to RL and how RL can be guided and shaped for policy improvement including speeding up initial convergence. We preliminary applied the model to a dynamic motion control problem and a goal-directed behavior control problem [MKWM13]. While the model improves control policy, it has a difficulty to solve partially observable Markov decision process scenarios. Thus, we further investigated and extended the learning model by introducing dynamics memories (like, temporal memories) into it. Reservoir computing was used to implement the dynamic memories. Besides, we also introduced reward modulated Hebbian learning into the combinatorial learning. As a result, the combinatorial learning with reservoir computing and reward modulated Hebbian Plasticity allows for the effective learning of goal directed behaviors in both fully and partially observable cases [DWMM13]. The developed learning framework will be applied to, e. g., the Kuka LWR arm or the humanoid robot ARMAR III for learning of arm pointing and reaching with obstacle avoidance as goal-directed control problems.

References

[1] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter. Learning the semantics of object-action relations by observation. *The International Journal of Robotics Research*, 32:951–970, 2011.

[2] M. Deniša and A. Ude. Discovering new motor primitives in transition graphs. In *Intelligent Autonomous Systems 12*, Advances in Intelligent Systems and Computing, vol. 192, pages 219–230. Springer, 2013.

[3] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrčen, A. Agostini, and R. Dillmann. Object-action complexes: Grounded abstractions of sensorimotor processes. *Robotics and Autonomous Systems*, 59:740–757, 2011.

[4] K. Yamane, M. Refvi, and T. Asfour. Synthesizing object receiving motions of humanoid robots with human motion database. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1629–1636, Karlsruhe, Germany, 2013.

Attached Papers

[DPAU13] Miha Deniša, Tadej Petrič, Tamim Asfour, and Aleš Ude. Synthesizing compliant reaching movements by searching a database of example trajectories. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 540–546, Atlanta, Georgia, 2013.

[DWMM13] S. Dasgupta, F. Wörgötter, J. Morimoto, and P. Manoonpong. Neural combinatorial learning of goal-directed behavior with reservoir critic and reward modulated hebbian plasticity. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 993–1000, Manchester, UK, 2013.

[MKWM13] P. Manoonpong, C. Kolodziejski, F. Wörgötter, and J. Morimoto. Combining correlation-based and reward-based learning in neural control for policy improvement. *Advances in Complex Systems*, 16(02n03), 2013.

[WSA$^+$13] Mirko Wächter, Sebastian Schulz, Tamim Asfour, Eren Aksoy, Florentin Wörgötter, and Rüdiger Dillmann. Action sequence reproduction based on automatic segmentation and object-action complexes. In *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 189–195, Atlanta, Georgia, 2013.

2013 13th IEEE-RAS International Conference on
Humanoid Robots (Humanoids).
October 15 - 17, 2013. Atlanta, GA

Synthesizing Compliant Reaching Movements by Searching a Database of Example Trajectories

Miha Deniša[1], Tadej Petrič[1], Tamim Asfour[2], and Aleš Ude[1,*]

Abstract— **We address the problem of generating new compliant reaching movements by searching a structured database of example trajectories. The proposed control framework is a multi-step process, where in the first step a human tutor teaches the robot how to perform a set of example reaching movements. In the second step, the recorded motion trajectories are executed with different velocities using a high gain feedback controller, for the purpose of learning corresponding torque control signals. The commanded torques are measured and stored together with the trajectory data. This data is organized in a hierarchical, graph-like structure, thereby providing the basis for search for new compliant trajectories, which can consist of parts of the previously acquired example movements. The proposed approach can construct a complete representation for newly discovered movements, including the feedforward torque commands. Finally, in the last step, the motion is executed using a low gain feedback controller and the associated feedforward torque signal. This ensures sufficient tracking accuracy and at the same time compliant behavior, which allows smooth interaction with the environment and is safe for cooperative task execution with humans. The usefulness of the proposed method was shown on a Kuka LWR robot.**

I. INTRODUCTION

A well established approach for dynamic robot control is the use of inverse dynamic models [1]. However, due to the increasing complexity of robot mechanisms such as humanoid robots, the accurate dynamical models are often difficult to obtain. To fulfil the gap, algorithms for machine learning were adopted in robotics because of their ability of learning complex models. Although learning algorithms became powerful enough to learn even the inverse dynamics [2], they still require a large amount of data for learning. As an alternative, different biology inspired methods were proposed for dynamic robot control. An extensive review of computational mechanisms for sensorimotor control, which covers methods from optimal feedback control [3] to the forward models and predictive control [4], was recently published by Franklin and Wolpert [5].

Inspired by the human sensorimotor ability, which can learn arbitrary dynamic tasks, we propose a new control framework that is based on learning a task dependent trajectory with corresponding control signals. The proposed framework is a multi step process, where in the first step, human tutor teaches the robot how to perform the desired task, e. g. a reaching movement. In the second step, the corresponding task dependent control signals are learned by executing the movement using a high gain feedback control loop, which ensures sufficient tracking accuracy. In the last step, the desired reaching movement is executed using the feedforward task dependent control signal and low gain feedback loop, which ensures compliance and stability. Because of the feedforward compensation, the robot will perform the desired task with similar accuracy as in the second step. Due to the low feedback gain, the robot exhibits a compliant behavior with low perturbation rejection, therefore it is safer for humans to work with.

However, just building a database of movements with the associated control signals might not be an optimal solution. It is not feasible to obtain all the necessary movement trajectories and their task dependent control signals for the entire workspace based on user demonstrations only. In this paper we propose to augment the set of available movement primitives by a hierarchical database search. A set of movement trajectories, which partly share a similar course of movement, can be used to discover new movements. Such an approach can significantly reduce the teaching effort, since a database that contains various example motions with similar sections can be used to generate new, not previously demonstrated movement trajectories. Through the hierarchical database search, new behaviors can be discovered, generated and eventually added to the database.

The proposed hierarchical search for new movement primitives is based on the work done mainly in the computer graphics community, which has long studied how to utilize large databases of diverse movements. This is in contrast to most of the work done in robotics, which is primarily focused on learning from a single demonstration [6] or learning from multiple demonstrated variations of the same type of movement [7]. It was shown that by organizing movements in motion graphs, smooth transitions between interconnected full-body movements can be found [8]. This approach was applied by Kovar et al. [9] to generate different styles of locomotion along arbitrary paths. In robotics, a graph-based representation similar to motion graphs was used by Yamane et al. [10]. They combined transition graphs with a binary tree database in order to generate human body locomotions. This research was later expanded to plan object receiving motions [11].

[*]The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273, Xperience, and from the Slovenian Research Agency under grant agreement no. J2-4284.

[1] M. Deniša, T. Petrič, and A. Ude are with the Jožef Stefan Institute, Department of Automatics, Biocybernetics and Robotics, Humanoid and Cognitive Robotics Lab, Ljubljana, Slovenia miha.denisa@ijs.si, tadej.petric@ijs.si, ales.ude@ijs.si

[2] T. Asfour is with the Karlsruhe Institute of Technology, Institute for Anthropomatics, High Performance Humanoid Technologies Lab, Karlsruhe, Germany asfour@kit.de

Besides in Yamane et al. [10], movement generation from trajectory libraries has also been investigated in [12], [13], [14], [15]. Like in our work, graph search is utilized in [12], but this work is focused on locomotion. Ude et al. [13] used variations of the same primitive movement to generate a new instantiation of a dynamic movement primitive that is optimal for a given situation. Like in [13], a situation descriptor is used also in [14] to transfer previously optimized trajectories to new situations. Mülling et al. [15] studied the problem of mixing dynamic movement primitives to generate optimal striking movements. The work described in this paper is most closely related to [10], [11] because our focus is on how to generate new dynamic movement primitives from parts of previously acquired example trajectories.

The rest of the paper is organized as follows. In Section II an approach for compensating robot dynamics is presented. It consists of Section II-A describing task trajectory learning, Section II-B describing learning of torque control signals, and Section II-C that deals with the learned trajectory execution. Section III proposes an approach for discovering new movement primitives thorugh hierarchical graph search. This is accomplished by building a database (Section III-A) and then using it to find and synthesize new compliant movements (Section III-B). The paper concludes with experimental results and conclusions.

II. COMPENSATION OF ROBOT DYNAMICS

To compensate for robot dynamics, we normally apply a generic approach which is based on inverse dynamical model of the robot. Assuming that the robot consists of rigid bodies, the joint space equations of motion are given by

$$\mathbf{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\varepsilon}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}) = \boldsymbol{\tau}, \tag{1}$$

where $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ are the joint positions, velocities and accelerations, respectively, $\mathbf{H}(\boldsymbol{q})$ is the inertia matrix, $\mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$ are the Coriolis and centripetal forces, $\boldsymbol{g}(\boldsymbol{q})$ are the gravity forces and $\boldsymbol{\varepsilon}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$ are the nonlinearities not considered in the rigid body dynamics, e. g. friction. We denote the inverse dynamic model of the robot (1) as $\mathbf{f}_{\text{dynamic}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$. Using the inverse dynamic model, a possible control approach for tracking the desired joint positions $\boldsymbol{q}_d$ is given by

$$\boldsymbol{\tau}_{cmd} = \mathbf{K}(\boldsymbol{q}_d - \boldsymbol{q}) + \mathbf{D}(\dot{\boldsymbol{q}}) + \mathbf{f}_{\text{dynamic}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}}), \tag{2}$$

where $\mathbf{K}$ is the diagonal matrix for stiffness and $\mathbf{D}(\dot{\boldsymbol{q}})$ is the damping term. Note that high values in matrix $\mathbf{K}$ stiffen the robot, which results in better tracking and error rejection in case of perturbations. On the other hand, if the stiffness values are low, the robot is compliant but the tracking accuracy might be poor.

To have both advantageous properties, i. e. accurate tracking and compliance, we proposed a new multi-step control system which includes feedforward torque signal that corresponds to the desired trajectory. Essentially, it is a pre-generated internal model-based control system. However, instead of using a complete inverse dynamic model for compensating the robot dynamics as usually, we use a set of layers based on dynamic movement primitives, which encode the information of the torque control signals alongside with the desired path (motion trajectories). The main advantage of the proposed control system is that is model free, i. e. the dynamic model of the robot is not needed. Moreover, since torque signal that corresponds to the desired task are feedforward during the execution step, the high tracking accuracy and natural compliant behavior are achieved. Natural compliance is the compliance of the mechanism itself. The proposed control system ensures that the robot is always compliant during the execution of the task, thereby ensuring that the collision contact forces are small and therefore the robot can perform tasks in unstructured environment and safely interact with humans.

A. Learning task trajectories

In the first step, the goal is to learn the motion trajectories (positions) demonstrated by a human teacher. Different techniques exist for teaching a desired motion to the robot; one can use kinesthetic guiding [16], haptic interfaces [17], motion capture systems [18], [19], etc. Kinesthetic guiding was used in this paper.

To encode motion trajectories, we use Dynamic Movement Primitives (DMPs). They are summarized in [6]. The equations below are valid for one degree of freedom (DOF). For multiple DOFs the equations can be used in parallel. For one DOF they are defined by the following nonlinear system of differential equations

$$\tau_{dmp}\dot{v} = \alpha_z(\beta_z(g-y)-v)+f(x), \tag{3}$$

$$\tau_{dmp}\dot{y} = v. \tag{4}$$

The linear part of Eq. (3) – (4) ensures that y converges to the desired final configuration, here denoted as g. The nonlinear part $f(x)$ modifies the shape of the movement and is defined by a linear combination of radial basis functions

$$f(x) = \frac{\sum_{i=1}^{N} w_i \psi_i(x)}{\sum_{i=1}^{N} \psi_i(x)} \tag{5}$$

$$\psi_i(x) = \exp(-h_i(x-c_i)^2), \tag{6}$$

where ψ_i defines the basis functions with centers at c_i and widths $h_i > 0$. As seen in Eq. (5), $f(x)$ is not directly time dependent. Instead, phase variable x defined in Eq. (7), with initial value $x(0) = 1$, is used to make the dependency more implicit:

$$\tau_{dmp}\dot{x} = -\alpha_x x \tag{7}$$

The phase is common across all DOFs. By specifying the time evolution through phase, it becomes easier to stop the clock in case of external perturbations, which cause the robot to deviate from the desired trajectory. It can be shown that – given the properly defined constants α_z, β_z, τ_{dmp}, $\alpha_x > 0$ – the above system is guaranteed to converge to the desired final configuration g.

We can encode demonstrated trajectories as DMPs by applying locally weighted regression and learn the target function defined as

$$f_j(t) = \tau_{dmp}\ddot{q}_j(t) + \tau_{dmp}\alpha_y\dot{q}_j(t) - \alpha_y\beta_y(g - q_j(t)), \tag{8}$$

where $q_j(t)$ denotes the demonstrated trajectory of the j-th joint at time t.

B. Learning torque control signals

In the second step we encode corresponding control torque signals for the kinematic trajectory $\boldsymbol{q}_d$, which was learned in the first step. To obtain the corresponding torque signals, we employed a high gain feedback controller, which ensured required tracking accuracy. The feedback control is given by

$$\boldsymbol{\tau} = \mathbf{K}(\boldsymbol{q}_d - \boldsymbol{q}) + \mathbf{D}(\dot{\boldsymbol{q}}), \tag{9}$$

Since kinematic trajectory is time invariant and the corresponding control torque signals must be time dependent, we introduce a task time multiplier κ that defines the duration of the task. With this in mind, DMP equations (3), (4), and (7) used for executing the demonstrated task trajectories while learning torque control signals, can be rewritten as:

$$\kappa\tau_{dmp}\dot{v} = \alpha_z(\beta_z(g-y)-v)+f(x), \tag{10}$$

$$\kappa\tau_{dmp}\dot{y} = v, \tag{11}$$

$$\kappa\tau_{dmp}\dot{x} = -\alpha_x x. \tag{12}$$

The equations for encoding and learning of the torque control signals are similar as given in section II-A. The main difference is that instead of learning the kinematic trajectory q_d, we learn the target function given by

$$f_j(t) = \kappa\tau_{dmp}\ddot{\tau}_j(t) + \kappa\tau_{dmp}\alpha_y\dot{\tau}_j(t) - \alpha_y\beta_y(g-\tau_j(t)) \tag{13}$$

where τ_j is the commanded torque signal for the j-th joint. By learning the control torque $\boldsymbol{\tau}_{ff}$, which is produced by the high gain feedback controller, the system essentially learns the corresponding inverse dynamics $\mathbf{f}_{\text{dynamic}}(\boldsymbol{q},\dot{\boldsymbol{q}},\ddot{\boldsymbol{q}})$ along the executed kinematic trajectory $\boldsymbol{q}_d$.

C. Executing the desired motion

In this step, the movement trajectory $\boldsymbol{q}_d$ and the corresponding torque control signal $\boldsymbol{\tau}_{ff}$ is executed, using a low gain feedback controller. The controller used for executing the motion is given by

$$\boldsymbol{\tau} = \mathbf{K}(\boldsymbol{q}_d - \boldsymbol{q}) + \mathbf{D}(\dot{\boldsymbol{q}}, C) + \boldsymbol{\tau}_{ff}, \tag{14}$$

where $\boldsymbol{\tau}_{ff}$ is the feedforward torque control signal, which was learned in the second step (note that feedforward torques are only active in the movement execution step). According to the control system analysis from [20], when feedforward models are used, a low-gain feedback loop is sufficient to preserve the stability of the system. On the other hand, without feedback loop the system would inevitably diverge, regardless of the precision of the feedforward model.

The main advantages of using feedforward models are the better tracking performance (compared to a system without feedforward terms) and the possibility to use low-gain feedback, which enables natural compliant behavior of the robot. Note that low-gain feedback has little impact on the mechanical (natural) compliance of the robot.

III. NEW TASK TRAJECTORIES

In the previous section we described how to acquire a set of desired trajectories with the corresponding torque control signals associated with the movement executions at different speeds. At this point the recorded trajectories can be played back using feedforward torque control signals. In this section we investigate how to combine the available trajectories to generate new trajectories with the corresponding feedforward torque control signals using hierarchical graph search. The application of hierarchical graph search for the generation of new robot movements as such is not new, see e. g. [10]. Here we study how to add the corresponding feedforward torque control signals to the newly found trajectories.

A. Building the database

To combine two different types of information in a hierarchical database, we use a similar approach as Yamane et al. [11]. Instead of storing and relating the motion of two subjects in an integrated database as in [11], we store and relate the kinematic trajectories $\boldsymbol{q}_d$ and the associated torque control signals $\boldsymbol{\tau}_{ff}$. The first part of the database thus represents the kinematic trajectories $\boldsymbol{q}_d$ obtained by kinesthetic guiding. We concatenate them in a sample position matrix:

$$\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n], \tag{15}$$

where $\boldsymbol{x}_i$ denotes the state vectors sampled at a given discrete time interval and n is the total number of all samples belonging to all learned kinematic trajectories incorporated into the database. State vectors are defined as

$$\boldsymbol{x}_i = [q_{1i}, \dot{q}_{1i}, q_{2i}, \dot{q}_{2i}, \ldots, q_{di}, \dot{q}_{di}]^T, \tag{16}$$

where j-th joint angle and its velocity at time t_i are denoted by q_{ji} and $\dot{q}_{ji}$, respectively, and d is the number of the robot degrees of freedom.

We use the sample joint matrix as a root node of a binary tree, which represents learned position trajectories. We use k-means algorithm (with $k=2$) to cluster similar state vectors and thus split the root node into two child nodes. The data in each of these nodes is then clustered again to gain the nodes at the next level of the binary tree, as shown in Fig. 1.

Criterion for when to stop splitting the tree nodes is based on the variability of data contained in the node. We define the mean distance d_k of node k as

$$d_k = \frac{\sum_{i=1}^{n_k} d(\boldsymbol{x}_{ki}, \boldsymbol{c}_k)}{n_k}, \tag{17}$$

where n_k denotes the number of state vectors clustered at node k. $d(\boldsymbol{x}_{ki}, \boldsymbol{c}_k)$ is the Euclidean distance between state vectors $\boldsymbol{x}_{ki}$ associated with node k and the node's centroid $\boldsymbol{c}_k$, which was calculated by the k-means algorithm. If d_k is lower than a predefined threshold, then state vectors contained in the node are similar and we stop splitting this node. With this we avoid the binary tree getting unnecessarily deep while ensuring the needed precision of the representation. With this criterion we cluster the data into nodes until we do not have any nodes left to split. To ensure that all state vectors are

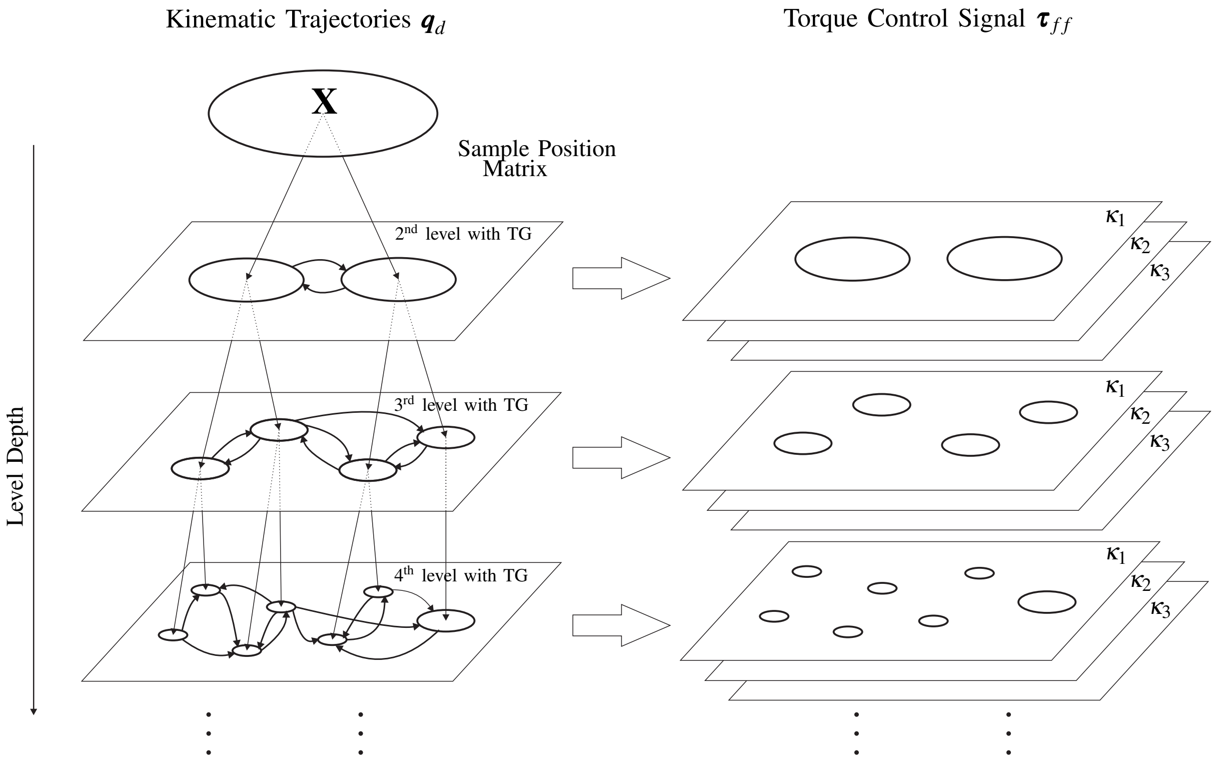


Fig. 1. Both parts of the database. The figure shows its structure, with kinematic trajectories represented on the left side and corresponding torque trajectories on the right. The sample joint matrix **X** is divided into two child nodes with *k*-means clustering. Then, the transition graph (TG), which represents probabilistic transitions between the nodes at this level, is built. The data associated with each node is clustered into child nodes for the 3rd level, where the TG is build again. We continue this procedure until all nodes fit the stopping criteria. Note that we expand those nodes to the last level and thus represent all of the data at all levels. The right side represents corresponding torques at different speed of execution i. e. different task time multipliers κ. At every level of the database, each node in the binary tree, on the left, has one or more (example figure shows three) corresponding torque means and time durations, on the right.

represented at all levels of the binary tree, every branch is extended to the last level.

Transition graph, representing all possible transitions between the nodes, is built at each level of the tree (see Fig. 1). The edge weights in the transition graph represent the probability of transition from one node to another. Transition probability from node k to node l is estimated by

$$P_{kl} = \frac{m_{kl}}{n_k} \tag{18}$$

where m_{kl} denotes the number of transitions observed in all trajectories of the original data, i. e. the number of all state vectors clustered in node k that have a successor in node l.

For further processing it is not necessary to store all state vectors $\boldsymbol{x}_k$ at each node of the binary tree. Instead, only the mean of the corresponding state vectors $\bar{\boldsymbol{x}}_k$ is stored at each node. If the node contains exactly one start/final configuration, we store it instead of the mean. In this way we ensure that movements generated by graph search end in the same end points as the learned position trajectories. In this step the time component is lost. We explain in Section III-B how time duration is estimated.

As we are synthesizing new task trajectories consisting of kinematic trajectories $\boldsymbol{q}_d$ and control torques $\boldsymbol{\tau}_{ff}$, the database must also encode torque control signals. We do not separately cluster the torque signals, but rather associate the torques with the corresponding nodes in the transition graph. This means that for each part of the kinematic trajectories, represented in a single node through the mean of state vectors, we store the corresponding means of torque signals $\bar{\boldsymbol{\tau}}_{ff}$ and time durations t_d. We do this multiple times as we execute the same movements with different time duration, i. e. with different task time multipliers κ. See Fig. 1 for example representation of the whole database.

B. Searching for new task trajectories

We start the search for new trajectories by selecting the desired start and end points on two different trajectories. In addition, the desired task time multiplier κ is selected. A binary tree level also needs to be selected. As the level determines the fidelity of reproduction compared to the original trajectories, we normally select the last level of the binary tree. We then try to find a path between the nodes corresponding to the desired start and end joint position. To achieve that we employ A* search algorithm in the transition graph at the desired level. As long as the two trajectories share a similar enough part, the most probable path is found and with it a sequence of nodes i. e. mean state vectors.

Based on the selected time multiplier κ, we add the corresponding mean torques $\bar{\boldsymbol{\tau}}_{ff}$ to the state vectors $\bar{\boldsymbol{x}}_k$ of the

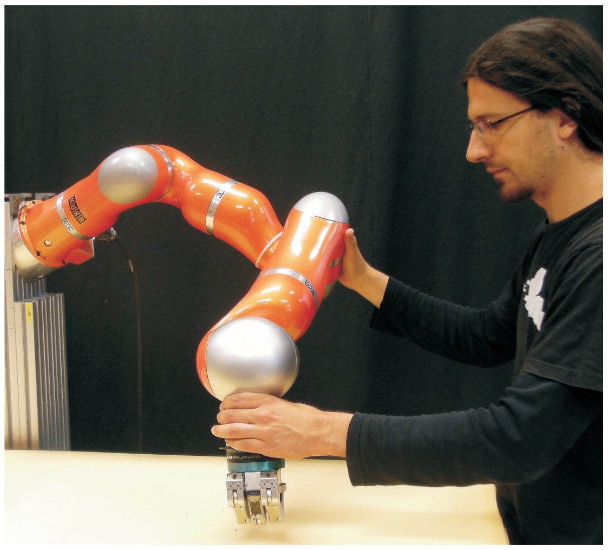


Fig. 2. Learning task trajectories by kinesthetically guiding the Kuka LWR robot arm.

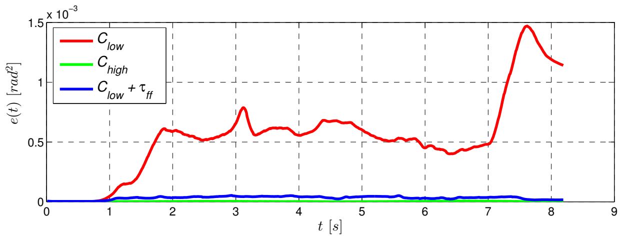


Fig. 3. The sum of all joint's tracking errors as defined in (20). The green line represents tracking errors while executing the movement with a high gain feedback controller (C_{high}). Red line represents tracing error while using a low gain feedback controller (C_{low}). Finally, the blue line represents the sum of all joint's tracking errors while using the proposed controller, low gain feedback controller and feedforward torque signals ($C_{low} + \tau_{ff}$). We can observe low tracking errors gained by incorporating feed-forward torque signals. Note that the gain used for feed-back loop controller C_{low} was 20 times lower than C_{high}.

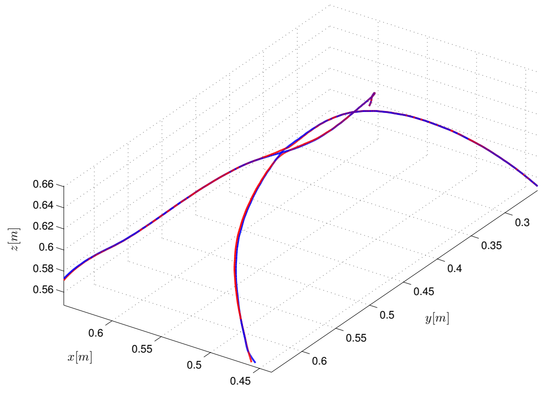


Fig. 4. The closest demonstrated and newly synthesized kinematic trajectories in task space. Blue lines represent demonstrated trajectories, while red lines represent newly synthesized kienmatic trajectories.

discovered sequence. We enhance this sequence further with time durations t_d corresponding to the added torques. Now we have a sequence of joint positions, torques and their time durations. The newly discovered sequence can be written as:

$$\left\{(\overline{\boldsymbol{x}}_1, \overline{\boldsymbol{\tau}}_{ff1}, 0), \left(\overline{\boldsymbol{x}}_2, \overline{\boldsymbol{\tau}}_{ff2}, \frac{t_{d1}+t_{d2}}{2}\right), \dots \right.$$
$$\left. \dots, \left(\overline{\boldsymbol{x}}_{n_p}, \overline{\boldsymbol{\tau}}_{ffn_p}, \frac{t_{d(n_p-1)}+t_{dn_p}}{2}\right)\right\}, \quad (19)$$

where n_p denotes the number of nodes on the trajectory.

We synthesize a trajectory from each discovered sequence by encoding it as a DMP. The DMPs describing newly synthesized kinematic trajectories and the corresponding torque control signals (also encoded as DMPs) can then be used to execute new, not directly shown, movements while remaining compliant, as described in Section II-C.

IV. EVALUATION

We evaluated the proposed approach using a Kuka LWR robot arm. The demonstrator taught the robot several reaching movements while kinesthetically guiding the arm (see Fig. 2). Two of the learned movements that intersect each other are shown in Fig. 4. All kinematic trajectories were encoded as DMPs. They were used in the second step to obtain corresponding torque control signals, as described in Section II-B. Each movement was executed three times with different task time multipliers $\kappa = \{1,2,3\}$. The learned movement trajectories $\mathbf{q}_d$ and the corresponding torque control signals $\boldsymbol{\tau}_{ff}$ were then used to execute the learned reaching movements using a low-gain feedback controller (14). Fig. 3 shows an example sum of all joint's tracking errors with respect to time

$$e(t) = \sum_{j=1}^{7} (q_{aj}(t) - q_j(t))^2, \quad (20)$$

where $q_{aj}(t)$ denotes the actual j-th joint position. We can observe that by using a low-gain feedback loop without the feed forward torque signal $\boldsymbol{\tau}_{ff}$ in order to achieve compliance, tracking error escalates greatly (red line) in comparison to a high-gain feedback loop (green line). However, by adding the feedforwad torque control, similar compliance can be achieved, while successfully tracking the desired trajectory $\boldsymbol{q}_d$ (blue line). Note that the low gain was 20 times lower than during the execution with a high-gain feedback loop.

Both the learned kinematic trajectories and the corresponding torque signals were used to build the database, as described in Section III-A. This database was used to find new reaching movements as described in section II-B. A* search algorithm found new sequences of nodes, as the demonstrated trajectories had parts that were sufficiently similar. Each new sequence started in the first node of one of the demonstrated trajectories and ended in the final node of one of the others. Each new sequence of mean position $\overline{\boldsymbol{x}}$ was then enhanced with mean torques $\overline{\boldsymbol{\tau}}_{ff}$ and corresponding time duration t_d three times, once per task time multiplier κ. Using DMPs we found complete representations of new

TABLE I

MEAN TRACKING ERRORS AND STANDARD DEVIATIONS FOR EACH JOINT AND THEIR SUMS WHEN EXECUTING REACHING MOVEMENTS. $Task_1$ AND $Task_2$ DENOTE TWO EXAMPLES OF THE DEMONSTRATED MOVEMENTS, WHILE $Task_{12}$ AND $Task_{21}$ DENOTE TWO EXAMPLES OF NEWLY SYNTHESIZED REACHING MOVEMENTS. ALL VALUES ARE IN $[10^{-5} rad^2]$.

$Task_1$ $(\kappa = 1)$	q_1	q_2	q_3	q_4	q_5	q_6	q_7	Σ
C_{high}	0.306 (0.289)	1.18 (1.69)	0.660 (0.491)	1.89 (1.65)	0.696 (1.23)	0.211 (0.292)	0.693 (1.09)	5.64 (6.74)
C_{low}	112 (101)	69.9 (77.7)	48.7 (28.3)	41.9 (98.0)	27.5 (44.9)	56.9 (145)	7.29 (11.6)	364 (507)
$C_{low} + \tau_{ff}$	2.51 (1.96)	21.7 (24.9)	1.54 (1.25)	5.49 (6.67)	3.31 (4.90)	1.79 (2.61)	5.55 (9.32)	41.9 (51.7)

$Task_2$ $(\kappa = 1)$	q_1	q_2	q_3	q_4	q_5	q_6	q_7	Σ
C_{high}	0.325 (0.235)	0.617 (0.938)	0.930 (0.989)	0.200 (0.292)	0.191 (0.197)	0.0836 (0.143)	0.405 (0.552)	2.75 (3.35)
C_{low}	100 (74.8)	48.5 (75.4)	31.3 (25.7)	38.1 (47.8)	9.41 (14.2)	12.3 (22.0)	4.94 (8.46)	245 (268)
$C_{low} + \tau_{ff}$	2.77 (2.93)	21.7 (26.5)	2.23 (2.53)	4.12 (7.43)	1.58 (1.90)	2.29 (3.38)	2.40 (3.47)	37.1 (48.2)

$Task_{12}$ $C_{low} + \tau_{ff}$	q_1	q_2	q_3	q_4	q_5	q_6	q_7	Σ
$\kappa = 1$	29.3 (44.4)	18.6 (22.6)	14.0 (24.6)	7.12 (9.87)	9.32 (12.5)	2.98 (3.26)	1.31 (2.25)	82.7 (119)
$\kappa = 2$	27.2 (61.5)	8.88 (7.04)	3.40 (9.56)	8.01 (12.2)	15.9 (21.8)	4.28 (4.89)	2.07 (2.21)	69.8 (119)
$\kappa = 3$	26.7 (68.4)	17.0 (15.7)	3.76 (7.74)	6.49 (8.97)	17.8 (24.6)	3.02 (3.71)	1.23 (1.66)	76.0 (131)

$Task_{21}$ $C_{low} + \tau_{ff}$	q_1	q_2	q_3	q_4	q_5	q_6	q_7	Σ
$\kappa = 1$	7.29 (6.42)	26.4 (19.9)	3.62 (3.30)	7.24 (9.59)	6.82 (11.5)	4.53 (4.60)	1.11 (2.52)	57.0 (57.8)
$\kappa = 2$	3.63 (3.86)	10.2 (8.01)	1.02 (1.80)	8.97 (13.7)	8.62 (11.77)	8.90 (3.60)	1.74 (1.67)	43.1 (44.3)
$\kappa = 3$	2.16 (3.07)	21.1 (17.0)	1.74 (1.93)	6.97 (11.5)	8.67 (11.8)	5.86 (4.51)	1.21 (1.48)	47.7 (51.3)

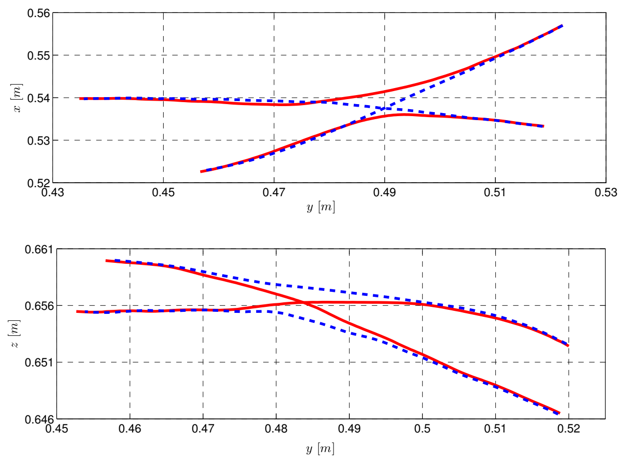


Fig. 5. Sections of the closest demonstrated trajectories and the newly synthesized trajectories in the task space. These sections represent transitions of new trajectories, represented with red lines, from one demonstrated trajectory to the other, represented with dashed blue lines. We can observe smooth and continuous transitions.

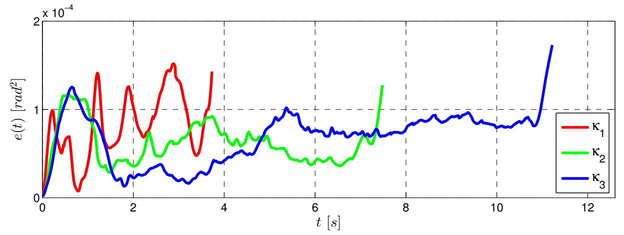


Fig. 6. The sum of all joint's tracking errors (20) of newly synthesized tasks. Errors of three example tasks are given. They were executed with different task time multipliers κ. Even though these movements were never directly shown, the tracking error remains within a tolerable range.

reaching movements trajectories. Two new example position trajectories in task space can be seen in Fig. 4, marked with red lines. For clarity, sections of demonstrated and new kinematic trajectories in two different 2D spaces are shown in Fig. 5.

All new reaching movements were executed on the robot as proposed in Section II-C. Fig. 6 shows joint position tracking errors of three example movements with different task time multipliers. Tracking errors are again defined as in Eq. 20. Tracking errors of newly synthesized tasks remain similar to those of directly demonstrated tasks.

Table I contains means and standard deviations of tracking errors $(q_{aj}(t) - q_j(t))^2$ for individual joints j. All values are in $10^{-5} rad^2$. First two parts of the table ($Task_1$ and $Task_2$) show errors with respect to the execution of the two closest reaching movements. They were both executed with task time multiplier $\kappa = 1$. Errors for this part of the table were measured by performing reaching movements with high gain feedback controller (C_{high}), low gain feedback controller (C_{low}), and low gain feedback controller with feedforward torque signal ($C_{low} + \tau_{ff}$). Low gain used for all movements was 20 times lower than high gain. We can observe how the tracking error drastically increases when we apply low gain instead of high gain. But there is a significant drop in error when we add feedforward torque signals to low gain feedback control. The last two parts of the table contain errors measured when executing two examples of the newly synthesized reaching movements ($Task_{12}$ and $Task_{21}$) while using low gain feedback control with feedforward torque signals. Each movement was executed for all three task time multipliers κ. We can immediately observe the consistency of tracking error over different multipliers κ. Secondly, there is no significant increase in tracking error even though these movements were newly synthesized. We can thus conclude that newly synthesized reaching movements can be executed using low gain feedback control with feedforward torque signals.

V. CONCLUSIONS

We proposed and evaluated an approach to discover new reaching trajectories in a database of example trajectories and to learn the corresponding dynamics. By feedforwarding the associated torque control signals, we can execute the reaching movements with a high tracking accuracy while exhibiting compliant behavior without using a full dynamic model. We showed that new reaching movements can be generated from a library of kinesthetically guided example movements, as long as they have sufficiently similar partial trajectories. In our experiments we used the developed approach to acquire a library of several reaching movements. Each of the kinesthetically guided movements was executed at different velocities using high gain controller and the associated torque control signals were stored in the database together with the kinematic trajectory. New movements were found using graph search. Our evaluation showed that newly synthesized movements maintain the needed tracking accuracy and compliance even though they were built from parts belonging to different movements. With a sufficient number of example movements, we could execute any reaching movement with compliant behavior while maintaining accuracy, without the need for a full dynamical robot model.

REFERENCES

[1] L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators*, ser. Advanced Textbooks in Control and Signal Processing. London: Springer, 2000.

[2] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey." *Cognitive Processing*, vol. 12, no. 4, pp. 319–40, 2011.

[3] C. G. Atkeson and J. M. Hollerbach, "Kinematic features of unrestrained vertical arm movements." *The Journal of Neuroscience*, vol. 5, no. 9, pp. 2318–30, 1985.

[4] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control." *Neural Networks*, vol. 11, no. 7-8, pp. 1317–29, 1998.

[5] D. W. Franklin and D. M. Wolpert, "Computational mechanisms of sensorimotor control." *Neuron*, vol. 72, no. 3, pp. 425–42, Nov. 2011.

[6] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[7] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with Gaussian Mixture Models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[8] C. Rose, M. Cohen, and B. Bodenheimer, "Verbs and adverbs: multidimensional motion interpolation," *IEEE Computer Graphics and Applications*, vol. 18, no. 5, pp. 32–40, 1998.

[9] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," *ACM Transactions on Graphics*, vol. 21, no. 3, July 2002.

[10] K. Yamane, Y. Yamaguchi, and Y. Nakamura, "Human motion database with a binary tree and node transition graphs," *Autonomous Robots*, vol. 30, no. 1, pp. 87–98, 2010.

[11] K. Yamane, M. Revfi, and T. Asfour, "Synthesizing object receiving motions of humanoid robots with human motion database," in *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013, pp. 1621–1628.

[12] M. Stolle, H. Tappeiner, J. Chestnutt, and C. G. Atkeson, "Transfer of policies based on trajectory libraries," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, 2007, pp. 2981–2986.

[13] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-Specific Generalization of Discrete and Periodic Dynamic Movement Primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[14] N. Jetchev and M. Toussaint, "Fast motion planning from experience: trajectory prediction for speeding up movement generation," *Autonomous Robots*, vol. 34, pp. 111–127, 2013.

[15] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *International Journal of Robotics Research*, vol. 23, no. 3, pp. 263–279, 2013.

[16] D. Kushida, M. Nakamura, S. Goto, and N. Kyura, "Human direct teaching of industrial articulated robot arms based on force-free control," *Artificial Life and Robotics*, vol. 5, no. 1, pp. 26–32, 2001.

[17] P. Evrard, E. Gribovskaya, S. Calinon, A. Billard, and A. Kheddar, "Teaching physical collaborative tasks: object-lifting case study with a humanoid," in *2009 9th IEEE-RAS International Conference on Humanoid Robots*, Paris, France, 2009, pp. 399–404.

[18] A. Ude, C. G. Atkeson, and M. Riley, "Programming full-body movements for humanoid robots by observation," *Robotics and Autonomous Systems*, vol. 47, no. 2-3, pp. 93–108, June 2004.

[19] J. Babic, J. G. Hale, and E. Oztop, "Human sensorimotor learning for humanoid robot skill synthesis," *Adaptive Behavior*, vol. 19, no. 4, pp. 250–263, 2011.

[20] M. W. Spong and M. Vidyasagar, *Robot dynamics and control*. Wiley, 2008.

2013 13th IEEE-RAS International Conference on
Humanoid Robots (Humanoids).
October 15 - 17, 2013. Atlanta, GA

Action Sequence Reproduction based on Automatic Segmentation and Object-Action Complexes

Mirko Wächter[1], Sebastian Schulz[1], Tamim Asfour[1],
Eren Aksoy[2], Florentin Wörgötter[2] and Rüdiger Dillmann[1]
[1]Institute for Anthropomatics, Karlsruhe Institute of Technology
Adenauerring 2, 76131 Karlsruhe, Germany
[2]Bernstein Center for Computational Neuroscience, University of Göttingen
III. Physikalisches Institut, Friedrich-Hund Platz 1, 37077 Göttingen, Germany
{waechter,s.schulz,asfour,dillmann}@kit.edu, {eaksoye,worgott}@physik3.gwdg.de

Abstract— **Teaching robots object manipulation skills is a complex task that involves multimodal perception and knowledge about processing the sensor data. In this paper, we show a concept for humanoid robots in household environments with a variety of related objects and actions. Following the paradigms of Programming by Demonstration (PbD), we provide a flexible approach that enables a robot to adaptively reproduce an action sequence demonstrated by a human. The obtained human motion data with involved objects is segmented into semantic conclusive sub-actions by the detection of relations between the objects and the human actor. Matching actions are chosen from a library of Object-Action Complexes (OACs) using the preconditions and effects of each sub-action. The resulting sequence of OACs is parameterized for the execution on a humanoid robot depending on the observed action sequence and on the state of the environment during execution. The feasibility of this approach is shown in an exemplary kitchen scenario, where the robot has to prepare a dough.**

I. Introduction and Related Work

Robots already are versatile helpers in structured industrial applications, and in the future, will they increasingly be supposed to work also in human centered environments. If robots are expected to interact in an unstructured household instead of working in a well-known factory environment, problems become more complex: In order to fulfill requirements of everyday activities, a wide variety of complex questions have to be solved. The robot has to cope with unfamiliar situations and unknown options for interaction so that behavior and actions have to be highly adaptive. Consequently, common methods of robot programming are not directly applicable in the above mentioned scenario.

To address these problems, the concept of Programming by Demonstration (PbD) has become a common approach. It has developed rapidly since its origins in the mid-1980s. The work of Halbert [1], shows how to program a software system by example. During the 90ies, similar approaches transferred into the robotics domain were presented [2], [3], [4]. PbD is a technique that enables teaching a robot new skills and behaviors by demonstrating actions on concrete examples. It enables the robot to learn continuously from human observation in scenes of everyday life. Using parameterizable representations of the observed data allows applying the demonstration to new situations. Ultimately, after setting up the PbD System, advanced programming skills will be not any longer needed because also untrained users are able to teach new skills to the robot by demonstration.

The development of suitable action representations and algorithms for PbD has been a key research topic for the last decades. Neuroscientists, computer scientists and engineers alike have been working on relevant problems concerning this issue. Schaal et al. [5] discussed imitation learning as methodology for PbD from a computational point of view.

Ijspeert et al. suggested nonlinear dynamical systems for the representation of a demonstrated motion [6], called *dynamic movement primitives (DMP)*. Following this approach Gams et al. used a two-layered dynamical system that allows to extract both the frequency and the waveform of the demonstration signal to learn periodic tasks on a humanoid [7]. Ernesti et al. enrich these DMP formulation by extending the canonical system by one dimension using a two dimensional oscillator, which unifies the representation of a periodic movement and its transients [8]. Regarding various perturbations while execution, the basic formulation can be modified in additional ways. Two exemplary works in this direction are an approach that enables the generalization of DMPs to new situations using the available training movements and the goal of the task [9] and another approach using nonlinear dynamical systems with gaussian mixture models, which can respond immediately to perturbations encountered during the motion [10].

In contrast Ude et al. [11] suggested using b-spline wavelets as the representation of whole-body motion. Several approaches make use of Hidden Markov Models to learn and reproduce demonstrated actions [12], [13], [14].

Besides kinematic representations, further problems have to be addressed, in order to build a system capable of interacting in unstructured environments. Among them, the representation for the manipulation of objects in the environment is a crucial factor. In order create trajectories for executing these tasks, position and orientation of the manipulated object must be determined [15], [16].

Another key question is how to decompose the observed task into a set of sub-actions. Such a subdivision can then be utilized to formulate motion primitives for the execution on the robot [18], [19], [20], [21], or as shown by Kulic et al. [22] even for incremental learning new templates in real-time.

An important principle, known as affordance and elementary for the proposed approach, is the relationship between objects and particular actions. Each object of a certain type, has a quality, which allows performing only specific actions using these objects. A framework for the action-centered representation of these correlations at different levels of hierarchy is presented by the formulation of the Object-Action Complex (OAC) concept [17]. OACs describe how a robot has to perform an action with an object to achieve a given goal. They take several sensor channels on different levels into account, ranging from sensorimotor- to semantic level. Examples of the sensorimotor level are joint angles or forces acting on the tool center point and, on the semantic level, the label of an object. Further, all OACs have preconditions and a prediction function associated with them, that encodes the belief how actions will impact on the world changes. This prediction is called the effect of the OAC.

Another preliminary method for our work, called ”semantic event chain” (SEC) [23], [24], employs the spatial relation between objects for the subdivision of the demonstrated task. This approach makes use of the visual perception, more precisely, stereo and optical flow information. Hence, it is limited to demonstrations which can be reliable captured by image processing, in particular fulfill the conditions of image segmentation. The basic idea of the SEC is to extract the change of contact relations, i.e. all moments (*keyframes*) when any object comes into contact or loses contact with another object. Analogous to OACs, the method is based on the affordance principle, in particular the linking of objects and object relations to actions. Therefore, the linking of both methods is evident.

II. Overview

In this paper, a novel concept for the automatic adaptive reproduction of human demonstration on a robot is presented. The goal is to enable the reproduction of beforehand completely unknown complex tasks with multiple object interaction. Knowledge about these tasks is acquired by observation of the human demonstration and the involved objects. This observation is further processed to determine distinct sub-actions and object relations. These sub-actions are associated with Object-Action Complexes (OACs) [17], which are organized in a prior known library. These OACs represent basic object manipulation and interaction skills. The association is done by utilizing the observed world states to select OACs with matching preconditions and effects. Using these associations the robot can reproduce the before unknown action sequence.

III. Our Approach

In this section, we will present in detail the components of the proposed system for automatic adaptive action sequence reproduction. The proposed system mainly consists of three components (see Fig. 1): demonstration, representation and execution. The demonstration component is responsible for the acquisition and segmentation of motion data. The representation component contains the object-action complex library and the association of the segmentation with specific object-action complexes. The execution component provides the adaptive reproduction of the observed action sequence on a robot.

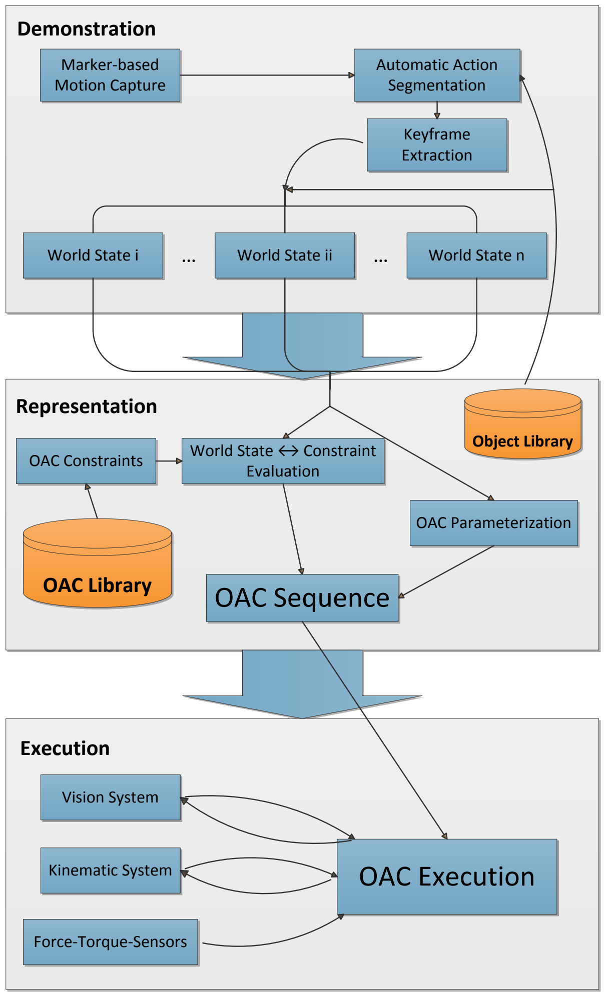


Fig. 1. System overview: The system consists of 3 components: demonstration, representation and execution.

A. Acquisition and segmentation of motion data

First, the human demonstration needs to be captured. The demonstrations are usually complex tasks like *preparing a dough*, which consist of several sub-actions. The trajectories of all components of the demonstrations need to be recorded. There are several ways to capture this data like inertial motion capture [25], marker-based motion capture with several cameras [26] or even 3D-based markerless motion capture [27]. Since this is a intensively researched field itself and outside the scope of this paper, a robust and precise marker-based motion capture system has been chosen in our experiments.

In order to extract the required trajectories of all components with this motion capture system, the agent and all involved objects have markers attached to them (see Fig. 2). To distinguish between the markers in the post-processing all markers are grouped by the object they are attached to. Further, the groups and the markers themselves are labeled. The labeling is important for the selection and parameterization of the object-action complexes, which are discussed in section III-B.

To segment the recorded action sequence we employ a method similar to the method presented in [23] by Aksoy et al. However, the main strategy remains the same. Instead of using a 3D vision system the demonstrated task is observed by an marker-based motion capture system. Consequently, the calculation of the object relations is not based on color segmentation but on the 3D Euclidian distance of objects over time. In this work, we call the resulting system for task segmentation *Automatic Action Segmentation* (AAS).

The environment is represented at any time for our approach as object relations. All object relations at a keyframe are considered as the *world state*. Contact changes between objects lead to a different world state. When the world state changes, there has been a change on the object relations, which in turn means that an action with an specific effect has happened. Thus, we are detecting actions by their effects on the environment. This approach is therefore model free in case of the actions and requires a simple spatial representation and a label for later processing for the objects. However, semantic information is not necessary.

The stated object relation is only of one kind at the moment: an object touches another object. One object can touch a set of other object, including the empty set. In contrast to the related work of Aksoy et al., we utilize the object distances to determine the keyframes instead of an exact graph-matching algorithm that extracts the main graphs of a sequence of graphs.

Further, Aksoy et al. use visual color segmentation and overlapping of color regions to detect changes of the object touching-relations. While this approach is flexible and does not require any prior knowledge about the objects, it lacks in robustness and precision. Therefore, we use marker trajectories from motion capture data to detect object touching-relations. All touching-relations at one frame are called the *world state*. The marker positions are the basis for all calculations. These markers are placed on the objects such that they are visible at all time. However, they do not represent the shape of the object. Thus, the distance calculations on the markers only may not detect all touching-relations between objects.

For calculation of the keyframes we use the markers $m_{G \in M, k=\{1 \dots |G|\}, i=\{1 \dots l\}} \in \mathbb{R}^3$ and the following constraints, where M is the set of marker group sets and l is the length of the trajectory:

Whenever

i) a marker $m_{G_1,k,i}$ is sufficiently close enough to a marker of another object $m_{G_2,j,i}$,
ii) and the change of distance $|m_{G_1,k,i} - m_{G_2,j,i}|'$ of two markers is sufficiently small for a minimum number of frames n,

the transition *non-touching* $\rightarrow$ *touching* is made:

$$\begin{array}{rl|l} & |m_{G_1,k,i} - m_{G_2,j,i}| < d & \exists n_0 \in \{1 \dots N\} \\ \wedge & |m_{G_1,k,i} - m_{G_2,j,i}|' < v & \forall i \in \{n_0 \dots n_0 + n\} \end{array} , \quad (1)$$

where n_0 is the frame with the *non-touching* $\rightarrow$ *touching* transition, constant d is the distance threshold, constant v is the distance-change threshold

The two markers are labeled as ”touching” from frame n_0 on until one of the previous conditions is false in the following *n* frames:

$$\begin{array}{rl|l} & |m_{G_1,k,i} - m_{G_2,j,i}| > d & \exists n_1 \in \{1 \dots N\} \\ \vee & |m_{G_1,k,i} - m_{G_2,j,i}|' > v & \forall i \in \{n_1 \dots n_1 + n\} \end{array} , \quad (2)$$

where n_1 marks the frame of the end of the touching-relation.

This way, passing other markers does not lead to touching-relations and noise in the data does not break up touching relations for a short time (see Fig. 3). At this point, for every change of the world state a new keyframe will be

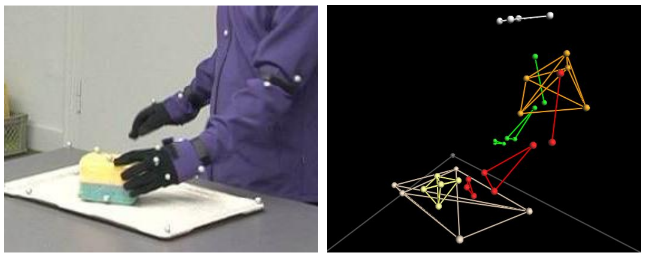


Fig. 2. Left: Human demonstrator with markers attached to him and all objects while wiping a tray. Right: The same frame of the demonstration as the 3D view of the reconstructed marker groups (different colors).

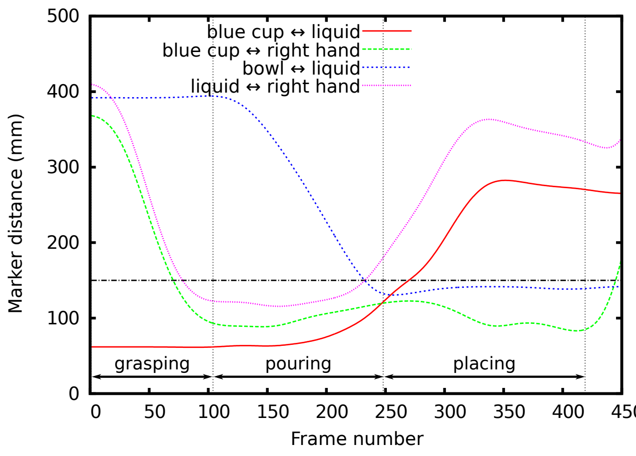


Fig. 3. Simple example of the AAS: depiction of the used measures (marker distance and change of distance) for the AAS. The black horizontal dotted line shows the distance threshold (*d*=150mm). The black vertical dotted lines show the resulting keyframes for the actions: grasping, pouring and placing.

inserted. However, this leads to oversegmentation of the action sequence with misleading keyframes. Some of these keyframes are separated by a relatively small number of frames, e.g. if an object is dropped onto another object. Since the demonstrated sub-actions in the scope of demonstrations for a robot always have a certain duration, these keyframes do not represent the desired segmentation. This can be solved by merging keyframes with only a few frames between them into one single keyframe. To achieve the merge the last keyframe of this group of keyframes and its relations are used. The previous keyframes of the group are discarded. A keyframe belongs to a group if the time difference to any other keyframe of the group is below a threshold. Proper chosing of this threshold is crucial for a correct segmentation. Too high and too low thresholds can both lead to additional false segments or missing segments depending on whether a touching relation started or ended.

With this segmentation method, the keyframes for the complete trajectory are calculated. In every keyframe, at least one object relation changes from *touching* to *non-touching* or vice versa. The corresponding object relations are stored for every keyframe and represent the current world state. In Fig. 3 the method's results are demonstrated in a simple example.

B. *Object-Action Complex (OAC) Library*

After applying the automatic action segmentation (see III-A), the demonstrated action sequence is subdivided into several sub-actions. However, the robot has no knowledge from the observation about how to reproduce the action sequence or any of the sub-actions. Although the trajectories of the markers and the involved objects are known, simply imitating the human demonstrator does not work. This is because the kinematics of the human and the robot usually are not interchangeable and because the objects are represented only by their attached markers. The robot has no knowledge about the shape of the objects and how to interact with them from motion capture data. Additionally, the used motion capture data does not contain any information about other perception channels, like the force applied during the demonstration. To enable the robot to reproduce the observed tasks a manually designed library of OACs is used. Though, only basic actions are stored in the library and complex tasks are observed.

The next step is to merge the gained information from the automatic action segmentation with the OAC library. The segmentation divides the action sequence in sub-actions, though, there is no association between the sub-actions and the OACs yet. Fortunately, the segmentation provides naturally for each sub-action a keyframe at the beginning and the end of the sub-action. At these keyframes, the current world state is stored. An OAC usually does not depend on the entire world state. Thus, a subset of the world states of the two keyframes is used as the preconditions and the effects of a sub-action.

There is a difference between the preconditions and effects of the OACs and two consecutive keyframes. The keyframes always contain instances of an object in the object relations, while on the other hand the OACs may contain variable terms in some or all the preconditions and effects, depending on the selected OAC. For instance, the grasping OAC has the following preconditions and effects:

$$\begin{array}{ll} Pre: & hand \leftrightarrow nothing \\ Effect: & hand \leftrightarrow object \in Graspable\ Objects \end{array}, \quad (3)$$

where $\leftrightarrow$ denotes a *touching* relation. Thus, to find the matching OAC, the object instances of the keyframes have to be validated against the compatibility with the OAC in question.

The world states of the segmented sub-action are utilized to perform a search within the OAC library to find a matching OAC by compairing the preconditions and effects of all OACs in the library with the previous and next world state of the segmented sub-action. It is important to notice, that OACs usually only depend on a small part of the world state and the remainder object relations are irrelevant. Hence, the object classes of the preconditions and effects of the OACs are checked against the specific object instances of the world states of the segmented sub-action for compatibility. If all preconditions and effects of an OAC are part of the segmented sub-action's world state, a link between the sub-action and the OAC is created and stored.

With the sequence of OACs it is possible to generate a new OAC that contains this sequence. The needed preconditions and effects of this new OAC can be calculated from the OAC preconditions and effects in the sequence. The complete method for this is shown in algorithm 1. The algorithm is divided into two parts: the calculation of the preconditions and the effects. The preconditions of sub-OACs are preconditions of the new OAC, if they are not effects of previous sub-OACs. The effects of the new OAC are the changes of the world state before the new OAC to the world state after it.

IV. Experiments

In this section, we will explain the experimental setup for the complete system and present an exemplary scenario for the application of the system.

A. *Experimental Setup*

The system for reproduction of the action sequence consist of two major hardware components. In this section, first the motion capture system and the second the humanoid robot, will be explained

1) Marker-based motion capture system: For motion capturing, we are using a multi-camera system with 10 cameras equipped with infrared lights and infrared filters. The human demonstrator has reflective markers attached to the torso, the arms and hands, and the head. On every object,at least three markers are asymmetrically placed to correct identify the pose and avoid instabilities in the assignment of markers.

2) Humanoid robot Armar-III: For the reproduction of the action sequence, we are using the humanoid robot ARMAR-III [28]. The kinematic chain of the robot consists of the following subsystems: As a base, it has a holonomic platform with three omniwheels. On this platform, a torso with three

```
          oacs := list of (p:preconditions, e:effects)
Input:    $w_s$ := world state before new OAC
          $w_e$ := world state after new OAC
Result: list of preconditions and list of effects for new
        OAC
foreach oac $o_{current}$ in sequence do
    $o \leftarrow o_{current}$
    $preconditionRequired \leftarrow true$
    foreach oac $o_{current}$ in sequence before $o$ do
        $o_b \leftarrow o_{current}$
        if $o_b.e \cap o.p \neq \{\emptyset\}$ then
            $preconditionRequired \leftarrow false$
        end
    end
    if $preconditionRequired = true$ then
        add $o.p$ to $p_{new}$
    end
end
foreach object $o_s$ in $w_s$ do
    foreach object $o_e$ in $w_e$ do
        if $o_s = o_e$ then
            add $(o_e.relations \setminus o_s.relations)$ to $e_{new}$
        end
    end
end
return $p_{new}$ and $e_{new}$
```

Algorithm 1: Calculation of preconditions and effects of new OAC

degrees of freedom (DOF) is placed. Like a human, it has two arms. Each of them consists of seven joints and has a five finger pneumatic hand attached to it. The head kinematics is divided into the neck joints with 3 DOF and the two eyes with a common tilt joint and independent pan joints, resulting in 10 DoF in total. The visual perception of the robot is accomplished with a foveal and a peripheral stereo vision system.

The robot is equipped with a 6D-force-torque-sensor in both wrists to measure the force applied to the hand. This sensor is used in most OACs as a state trigger, trajectory modifier to reduce applied forces or merely as a trigger for aborting the OAC as a safety precaution.

3) Environment: The experiment was conducted in a kitchen environment. The robot is standing at a table with several objects on it: two cups of different color, one mixing bowl and a mixer. The cups are in the demonstration filled with a big marker to symbolize the liquid. For the reproduction a robot friendly liquid replacement, i.e. small balls, are used.

B. Exemplary Scenario

Analogously to the environment, we chose a task that belongs to the kitchen scenario: preparing a dough. This a complex task consisting of several OACs with multiple objects involved. It requires and shows all the components of our approach.

The execution of the task by an human demonstrator while being observed by the motion capture system is realized by the following sequence of commands:

1) Pour the *liquid one* into the *orange bowl*
2) Pour the *liquid two* into the *orange bowl*
3) Use the *electric mixer* for mixing the *dough*

This is one possible description for the OAC sequence that probably would be sufficient for most humans. It is written in a way that some important data for the execution is not explicitly described. A human infers the missing data from the context. However, a robot cannot execute this plan since it has a different view on the actions *pouring* and *mixing*. The OACs used with this system focus on moments when objects touch each other or stop touching. Thus, the AAS extracts the segmentation that is shown in Fig. 4. The figure shows the whole process of reproduction of action sequence of the described exemplary scenario. The left column shows the demonstration by a human at the keyframes that are extracted by the AAS. The graphs in the middle column represent the world state at each keyframe. Each node of the graph depicts an object and the connections illustrate the touching-relations between objects. Black connection lines depict already existing relations, while red solid lines stand for a new touch-relation, and dotted lines for fading touching-relations. The grey boxes in the right column show the selected OAC with the matching constraints. The wildcards in the OAC constraints are filled with the specific instances for this action sequence, which result from the previous and next world state as illustrated with the black arrows. The pictures on the far right show the robot while executing the action sequence.

During the reproduction of the dough preparation the robot needed four different OACs, some of them multiple times: Grasping, pouring, placing and mixing. This led to a new OAC with the following preconditions:

- $Left\ hand \leftrightarrow Nothing$
- $Right\ hand \leftrightarrow Nothing$
- $Liquid\ One \leftrightarrow Red\ cup$
- $Liquid\ Two \leftrightarrow Green\ cup$

and the resulting effects:

- $Right\ hand \leftrightarrow Mixer$
- $Liquid\ One \leftrightarrow Orange\ bowl$
- $Liquid\ Two \leftrightarrow Orange\ bowl$
- $Mixer \leftrightarrow Orange\ bowl$

The new OAC is inserted into the OAC library and available for future executions.

V. Conclusion

In this paper, we presented a system that first enables robots to observe human interaction with objects in unstructured environments. It then decomposes demonstrated tasks into sub-actions that can be mapped onto the entries of an action library. Finally, action sequences are parameterized for the current situation and can be reproduced on a robot. The feasibility was shown in an exemplary scenario for preparing a dough.

It can be summarized that our approach performs well in the chosen scenario, which covers frequent actions in the kitchen domain. The following desirable features that are circumvented or just not supported by other approaches are natively supported by our approach: Due to the fact that AAS relies on world states instead of commonly used agent poses we achieved time-invariance and pose-invariance. Furthermore, it is invariant to the kinematics of the demonstrator.

These features empower our approach to reproduce tasks that were previously completely unknown to the robot by automatically splitting them up in known sub-actions. However, these sub-actions need to be known beforehand and are the backbone of this approach. Learning these sub-actions only from observation is, even for a human, a challenging task. Humans usually need to evaluate them before achieving complete comprehension and take at least the haptic perception into account as well. To teach the robot these sub-actions, more specialized approaches might be required, for instance a multimodal approach integrating several sensor channels.

Future work could concentrate on integrating an interactive sequence completion for unknown sub-actions, where the robot signals that he could not comprehend a demonstrated sub-action and asks for user interaction to help him understanding it. The segmentation algorithm, in particular the keyframe merging, could be extended through automatic adaptation of the hyperparameters to reduce the dependency on correct parameterization.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement № 270273 (Xperience).

REFERENCES

[1] D. C. Halbert, "Programming by example," Ph.D. dissertation, University of California, Berkeley, 1984.

[2] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching: Extracting reusable task knowledge from visual observation of human performance," *IEEE Transactions on Robotics and Automation*, vol. 10, pp. 799–822, 1994.

[3] S. Muench, J. Kreuziger, M. Kaiser, and R. Dillmann, "Robot programming be demonstration (rpd) – using machine learning and user interaction methods for the development of easy and comfortable robot programming systems," in *Proc. International Symposium on Industrial Robots (ISIR)*, 1994, pp. 685–693.

[4] H. Friedrich, S. Münch, R. Dillmann, S. Bocionek, and M. Sassin, "Robot programming by demonstration (rpd): supporting the induction by human interaction," *Mach. Learn.*, vol. 23, no. 2-3, pp. 163–189, 1996.

[5] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical Transactions of the Royal Society of London: Series B, Biological Science*, vol. 358, no. 1431, pp. 537–547, 2003.

[6] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *In IEEE International Conference on Robotics and Automation (ICRA2002)*, 2002, pp. 1398–1403.

[7] A. Gams, M. Do, A. Ude, T. Asfour, and R. Dillmann, "On-Line periodic movement and force-profile learning for adaptation to new surfaces," Nashville, USA, December 2010.

[8] J. Ernesti, L. Righetti, M. Do, T. Asfour, and S. Schaal, "Encoding of periodic and their transient motions by a single dynamic movement primitive," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2012, pp. 57–64.

[9] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *Robotics, IEEE Transactions on*, vol. 26, no. 5, pp. 800–815, 2010.

[10] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *Robotics, IEEE Transactions on*, vol. 27, no. 5, pp. 943–957, 2011.

[11] A. Ude, C. G. Atkeson, and M. Riley, "Programming full-body movements for humanoid robots by observation," *Robotics and Autonomous Systems*, vol. 47, pp. 93–108, 2004.

[12] S. Calinon, F. Guenter, and A. Billard, "Goal-directed imitation in a humanoid robot," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 299–304.

[13] S. Calinon and A. Billard, "Recognition and reproduction of gestures using a probabilistic framework combining pca, ica and hmm," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 105–112.

[14] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann, "Imitation learning of dual-arm manipulation tasks in humanoid robots," *International Journal of Humanoid Robotics*, vol. 5, no. 2, pp. 183–202, December 2008.

[15] A. Ude, "Trajectory generation from noisy positions of object features for teaching robot paths," *Robotics and Autonomous Systems*, vol. 11, no. 2, pp. 113–127, 1993.

[16] A. Ude, D. Omrčen, and G. Cheng, "Making object learning and recognition an active process," *International Journal of Humanoid Robotics*, vol. 5, no. 2, pp. 267–286, 2008.

[17] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrcene, A. Agostinig, and R. Dillmann, "Object-action complexes: Grounded abstractions of sensorimotor processes," *Robotics and Autonomous Systems*, 2011.

[18] J. Barbic, A. Safonova, J. Y. Pan, C. Faloutsos, J. K. Hodgins, and N. S. Pollard, "Segmenting motion capture data into distinct behaviors," in *GI '04: Proceedings of Graphics Interface 2004*. School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada: Canadian Human-Computer Communications Society, 2004, pp. 185–194.

[19] D. Kulic and Y. Nakamura, "Incremental learning of human behaviors using hierarchical hidden markov models," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 4649–4655.

[20] K. Yamane, M. Revfi, and T. Asfour, "Planning object receiving motions of humanoid robots with human motion database," in *ICRA*, 2013.

[21] M. Denisa and A. Ude, "Discovering new motor primitives in transition graphs," in *Intelligent Autonomous Systems 12*, ser. Advances in Intelligent Systems and Computing, S. Lee, H. Cho, K.-J. Yoon, and J. Lee, Eds. Springer Berlin Heidelberg, 2013, vol. 193, pp. 219–230.

[22] D. Kulic, W. Takano, and Y. Nakamura, "Online segmentation and clustering from continuous observation of whole body motions," *Robotics, IEEE Transactions on*, vol. 25, no. 5, pp. 1158–1166, 2009.

[23] E. E. Aksoy, A. Abramov, J. Dörr, N. Kejun, B. Dellen, and F. Wörgötter, "Learning the semantics of object-action relations by observation," *The International Journal of Robotics Research (IJRR)*, 2011.

[24] E. E. Aksoy, A. Abramov, F. Wörgötter, and B. Dellen, "Categorizing object-action relations from semantic scene graphs," in *ICRA*, 2010, pp. 398–405.

[25] D. Vlasic, R. Adelsberger, G. Vannucci, J. Barnwell, M. Gross, W. Matusik, and J. Popović, "Practical motion capture in everyday surroundings," *ACM Transactions on Graphics (TOG)*, vol. 26, no. 3, p. 35, 2007.

[26] J. Lee, J. Chai, P. S. Reitsma, J. K. Hodgins, and N. S. Pollard, "Interactive control of avatars animated with human motion data," in *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 491–500.

[27] P. Azad, *Visual Perception for Manipulation and Imitation in Humanoid Robots*. Springer, 2009, vol. 4.

[28] T. Asfour, K. Regenstein, P. Azad, J. Schröder, and R. Dillmann, "ARMAR-III: a humanoid platform for perception-action integration," in *Proc., International Workshop on Human-Centered Robotic Systems (HCRS), Munich*, 2006, pp. 51–56.

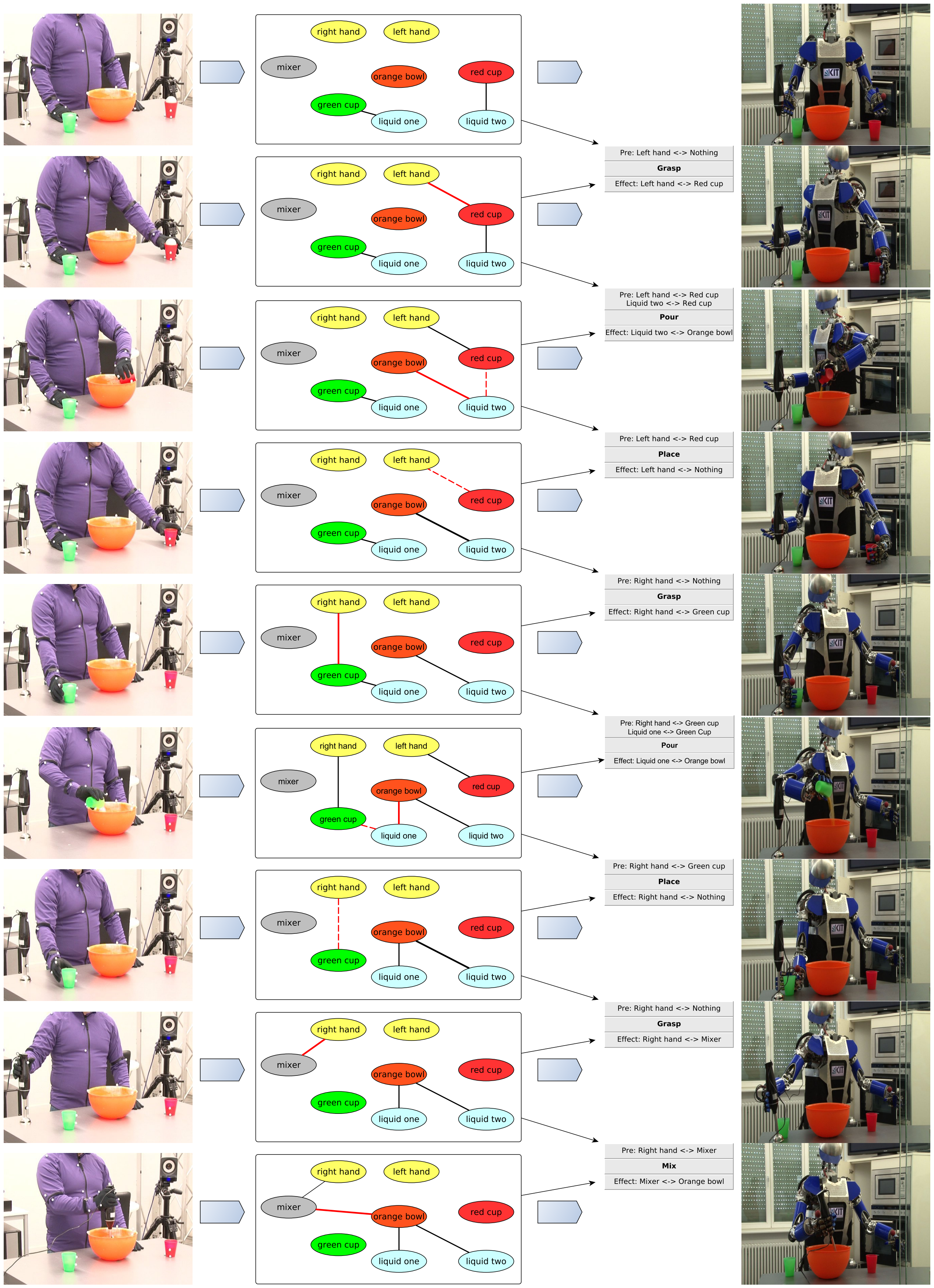

right hand
left hand
mixer
orange bowl
red cup
green cup
liquid one
liquid two
Pre: Left hand <-> Nothing
Grasp
Effect: Left hand <-> Red cup
Pre: Left hand <-> Red cup
Liquid two <-> Red cup
Pour
Effect: Liquid two <-> Orange bowl
Pre: Left hand <-> Red cup
Place
Effect: Left hand <-> Nothing
Pre: Right hand <-> Nothing
Grasp
Effect: Right hand <-> Green cup
Pre: Right hand <-> Green cup
Liquid one <-> Green Cup
Pour
Effect: Liquid one <-> Orange bowl
Pre: Right hand <-> Green cup
Place
Effect: Right hand <-> Nothing
Pre: Right hand <-> Nothing
Grasp
Effect: Right hand <-> Mixer
Pre: Right hand <-> Mixer
Mix
Effect: Mixer <-> Orange bowl

Fig. 4. Demonstrated action sequence at the detected keyframes (left column), the extracted world state at each keyframe (middle column), and the selected OACs from the two corresponding world states, which are executed on the robot (right column).

2013 IEEE International Conference on Systems, Man, and Cybernetics

Neural Combinatorial Learning of Goal-directed Behavior with Reservoir Critic and Reward Modulated Hebbian Plasticity

Sakyasingha Dasgupta[†], Florentin Wörgötter[†], Jun Morimoto[‡] and Poramate Manoonpong[†]
[†]Bernstein Center for Computational Neuroscience (BCCN), Georg-August-Universität,
Friedrich Hund Platz 1, 37077, Göttingen, Germany
dasgupta@physik3.gwdg.de
[‡]ATR Computational Neuroscience Laboratories, 2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288, Japan

***Abstract*—Learning of goal-directed behaviors in biological systems is broadly based on associations between conditional and unconditional stimuli. This can be further classified as classical conditioning (correlation-based learning) and operant conditioning (reward-based learning). Although traditionally modeled as separate learning systems in artificial agents, numerous animal experiments point towards their co-operative role in behavioral learning. Based on this concept, the recently introduced framework of neural combinatorial learning combines the two systems where both the systems run in parallel to guide the overall learned behavior. Such a combinatorial learning demonstrates a faster and efficient learner. In this work, we further improve the framework by applying a reservoir computing network (RC) as an adaptive critic unit and reward modulated Hebbian plasticity. Using a mobile robot system for goal-directed behavior learning, we clearly demonstrate that the reservoir critic outperforms traditional radial basis function (RBF) critics in terms of stability of convergence and learning time. Furthermore the temporal memory in RC allows the system to learn partially observable markov decision process scenario, in contrast to a memoryless RBF critic.**

Keywords*—Re-inforcement learning, Reservoir networks, Correlation learning, Temporal memory

I. Introduction

Operant conditioning (or reinforcement learning) and classical conditioning (or correlation-based learning) form the two classes of conditioning for associative learning in biological systems. Several animal experiments provide evidence of effective learning when these two classes are combined together [14]. Inspired by this in [8] the neural combinatorial learning framework was introduced. This combined the input correlation learning (ICO) [13] and actor-critic reinforcement learning (RL) [1] for controlling artificial agents in continuous time. The learning performance of the combined system clearly outperforms the individual mechanisms for both standard benchmark learning problems as well as complex goal-directed behavior problems. However, the actor-critic learner was modelled in a traditional manner, using a feedforward radial basis function (RBF) critic network [9]. Although this works well for most standard memoryless markovian learning tasks, it fails to approximate the value function in case of non-markovian or partially observable markov decision problems (POMDP).The role of the critic within the actor-critic learning paradigm is crucial as it needs to approximate the expected cumulative future reward (value function) such that the temporal difference (TD) error can be minimised. This TD-error in turn drives the policy of the actor and guides the behavior of the controlled agent. In case of highly non-linear environments where the agent has only partial sensory capabilities, a critic with temporal memory is required. As such in this paper, we replace the previous RBF based critic with a new recurrent neural network based adaptive critic of the reservoir computing (RC) type. RC networks [4][5] make use of a randomly connected dynamic reservoir with delayed temporal memory capacity [2]. Using a recursive least squares algorithm, this type of critic can be trained very fast in an online setup. Furthermore due to internal feedback connections, the short term memory of incoming sensory information can be used to solve POMDP learning problems. The RC based critic enhances the actor-critic based learner. In order to combine it with the ICO learning component of the combinatorial framework, learning of the connection weights between the two systems (Fig. 1) is very important. We solve this problem by introducing a new learning rule based on a biologically plausible mechanism called reward modulated Hebbian plasticity (RMHP) [6]. The RMHP rule updates the connection weights between ICO and actor-critic RL by checking for correlations between a constant reward signal and the deviation from the mean output level of the respective learning mechanisms. As such depending on the learner which drives the agent towards the correct goal (i.e. positively reinforced), the weight adaptation proceeds to finally find a suitable combination between the two learning systems.

Previously in [10][11] an application of the echo-state network (specific RC network) as an adaptive critic for reinforcement learning was presented. Although the authors implemented an online learner, the training and testing data for the RC network were carried out by manually controlling a wheeled robot. Moreover these implementations were designed with the purpose of minimizing a specific utility function for obstacle avoidance, door-passing scenario or very simple learning to reach a single goal. In contrast we implement a completely continuous learner where the reservoir critic learns online without any initial manual control of the robot. Furthermore, to the best of our knowledge, this is the very first implementation that combines correlation learning with reservoir based actor-critic learning and reward modulated Hebbian learning to succesfully

demonstrate a more efficient and fast learner. In addition, the RMHP learning rule is both biologically plausible and an effective mechanism to learn the contribution of competing systems modulated by constant reward signal. As proposed in our previous work [2] self-adaptation of the reservoir neurons non-linearity is carried out using a general intrinsic plasticity mechanism based on the Weibull probability distribution.
We test our combined network on a complex goal-directed behavior task with a simulated wheeled robot for both fully and partially observable scenarios. The RC based adaptive critic clearly outperforms feedforward critic networks based on RBF kernels, both in terms of stability of performance as well as the learning time. Moreover the RMHP based weight adaptation rule, by working on a very slow timescale is able to accurately combine the two learning systems in an adaptive manner. Specifically this type of a neural combinatorial learning framework based on reservoir critics can be used to solve complex control problems as well as to solve tasks with delayed reward or partially observable state space, in continuous time.
This article is organized as follows. Section II introduces the neural combinatorial learning framework in greater detail with descriptions of the new reservoir based actor-critic learner (section II A) and the reward modulated Hebbian plasticity rule (section II B). Section III presents the experimental setup with the discussion of results. This is followed by the conclusion in Section IV.

II. Neural Combinatorial learning framework

In this section we briefly describe the neural combinatorial learning framework (CLF), as introduced in our previous work [8]. The CLF combines two classes of associative learning, namely classical conditioning and operant conditioning, as a dual learning system. It is used for goal directed behaviors in continuous state-action spaces. Classical conditioning involves the presentation of two different stimuli often termed as a conditional stimulus (CS) and an unconditional stimulus (US), leading to corresponding responses. The agent learns the association between the US and CS such that after learning completes, it now responds to the CS rather than the original unconditioned response (an innate reflex action) to the US. In general the CS acts as a predictor signal (occuring earlier in time) for the US, e.g. the famous Pavlovian dog [12] initially salivates (unconditioned response) at the sight of food (US) and after learning salivates at the ring of a bell (CS) much prior to the sight of food. However this type of learning occurs in the absence of any explicit future positive or negative feedback (other than the immediate reflex signal) for a particular action. In contrast, Operant conditioning based learning involves an explicit reinforcer or reward signal that provides positive or negative feedback to the agent for every corresponding action. Over time the agent learns to respond with the desired action such that it maximises (for the positive case) the total accumulated reward. As such this type of conditioning is popularly termed as Reinforcement learning (RL).
Although these two mechanisms are distinct from each other they seem to occur in combination as suggested from several animal behavioral studies. For a more clearer understanding let us consider the example of Pavlov's dog once again. Say, once the bell is rung, the dog is now required to perform a specific task (e.g. stand on two legs) and only then it receives food. In this slightly modified scenario, the bell is still the conditional stimulus; however, now the food acts as the reinforcer or reward signal. The dog learns to associate the sound of bell and food and starts to salivate based on classical conditioning. Interestingly after sufficient repetitions, the dog would learn to perform the desired action of standing on two legs as soon as it hears the bell and expect to receive the food as reward. Thus the overall behaviour is shaped through a combined learning system.

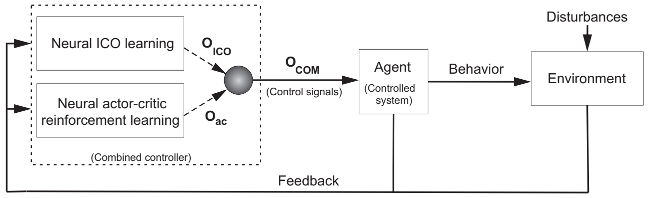


Fig. 1. Combinatorial learning framework with parallel combination of ICO learning and actor-critic reinforcement learning. Individual learning mechanisms adapt their weights independantly and then their final weighted outputs (O_{ico} and O_{ac}) are combined into O_{com} using a reward modulated Hebbian plasticity rule (dotted arrows represent plastic synapses). O_{com} controls the agent behavior (policy) while sensory feedback from the agent is sent back to both the learning mechanisms in parallel.

Inspired by such biological systems the CLF acts as a neural learning system that succesfully combines classical conditional (CC) with operant or reward based conditioning. Input correlation learning (ICO)[1] [13] was implemented as an example of CC, while a continuous actor-critic learner [1] was implemented as an example of reward based conditioning. Taking advantage of the individual learning mechanisms, the combined framework can learn the appropiate control policy for the agent in a fast and robust manner outperforming the singular implementation of the individual components.

The input correlation learning (ICO) and actor-critic RL subsystems can either be combined in series or in parallel. Previously in [7] serial combination was presented, where the ICO learner was used for reward related feature space extraction and provide prior knowledge to the actor-critic learner. Although this considerably improved the performance of the combined learning system, it suffered from the drawback of technical inconvinience of running the learning systems separately. This is also biologically less plausible. We extended this to a parallel combination in [8], however with a memoryless radial basis function critic network. Furthermore the subsystems were combined with equally weighted contribution in a non-adaptive manner to control the overall action of the Agent. As such in this work, we start with a parallel combination (Fig. 1) of the two individual learning systems. The actor-critic reward based learner is extended with a dynamic adaptive reservoir based critic with delay temporal memory capability [2] that can handle partially observable markov decision process problems (POMDP) in continuous time. Furthermore we implement a new reward modulated hebbian plasticity rule that learns the degree of contribution of the two learning systems.
The learning goal of the ICO learning system is to use a predictive signal (CS) in order to predict the occurence of the

[1]ICO learning is implemented as a differntial Hebbian learner. For more details refer to [8]

reflex signal (US). This in general enables the agent to react earlier and avoid the reflex altogether. Here the synaptic adaptation takes place by changes via heterosynaptic interactions as a consequence of the order of the arriving inputs. If the predictive inputs (agents sensory signals) are followed by the reflex input, the plastic synapses of the predictive inputs get strengthened and if the order is reveresed, it weakens based on a differential Hebbian learning method. For further details of the ICO learning system, the reader is refered to [13] [16].

A. Actor-critic Learning with Dynamic Reservoir

The continuous actor-critic reinforcement learning scheme is particularly suited for complex continuous state-action problems while at the same time being based on a biological learning model [3]. The basic learning model can be divided into two sub-mechanisms popularly termed as the actor and the adaptive critic (Fig. 2). The actor behaves as the main controller of an agent, while the critic provides an evaluative feedback or reinforcement signal to the actor by observing the consequences of its behaviour in the environment (controlled system). This evaluative feedback in general acts as a measure of goodness of behaviour i.e. overtime the agent learns to anticipate reinforcing events.
Inspired by the reservoir computing framework, here we use a large recurrent neural network (dynamic reservoir) as the critic. This provides a dynamic network with a large repertoire of reservoir signals that can be used to approximate the value function $v(t)$. It approximates the accumulated sum of the future rewards $r(t)$ with the discount factor γ where, $0 \leq \gamma < 1$.

$$v(t) = \sum_{i=1}^{\infty} \gamma^{i-1} r(t+i). \tag{1}$$

The primary goal of the critic is to predict $v(t)$ such that the temporal-difference error δ (TD-error) is minimized over time. The TD-error δ is computed from the predictions as follows:

$$\delta(t) = r(t) + \gamma v(t) - v(t-1). \tag{2}$$

The reservoir network (Fig. 2 bottom) is constructed as a random RNN with N internal neurons and fixed synaptic connectivity. The recurrent neural activity within the dynamic reservoir varies as a function of it's previous activity and the current driving input signal. As such, the discrete time state dynamics of reservoir neurons is given as:

$$\mathbf{x}(t+1) = (1-\lambda)\mathbf{x}(t) + \lambda f_{sys}(\mathbf{W}_{in}\mathbf{u}(t+1) + \mathbf{W}_{sys}\mathbf{x}(t)), \tag{3}$$

$$\mathbf{y}(t) = f_{out}(\mathbf{W}_{out}\mathbf{x}(t)), \tag{4}$$

where $\mathbf{x}(t)$ is the N dimensional vector of reservoir state activations, $\mathbf{u}(t)$ is the input to the reservoir, consisting of the agent's states (sensory inputs) and $\mathbf{y}(t)$ is the vector of output neurons. Here the predicted value function $v(t) = \mathbf{y}(t)$. The reservoir time scale is controlled by the parameter λ, where $0 < \lambda \leq 1$. $\mathbf{W}_{in}$ and $\mathbf{W}_{sys}$ are the input to reservoir weights and the internal reservoir recurrent connection weights, respectively.
The output weights $\mathbf{W}_{out}$ are calculated using the recursive least squares (RLS) algorithm at each time step, while the training inputs $\mathbf{u}(t)$ are being fed into the reservoir. $\mathbf{W}_{out}$ are calculated such that the overall TD-error is minimized. We implement the RLS algorithm using a fixed forgetting factor ($\lambda_{RLS} < 1$) as follows:

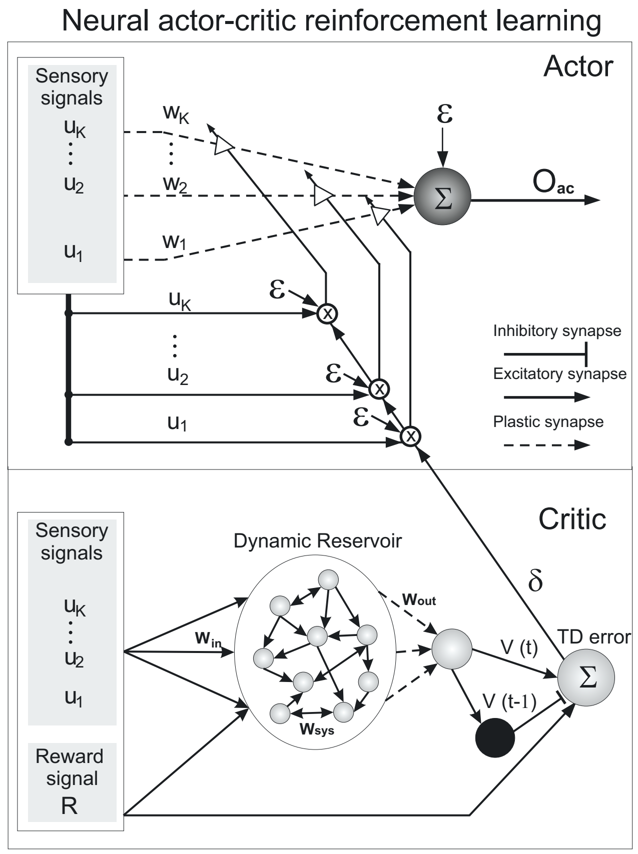


Fig. 2. The Neural circuit of actor-critic RL based on TD learning. (Top) The actor modeled as a stochastic neural network. (Below) The critic modeled using a dynamic reservoir network (details in text).

RLS algorithm for self-adaptive reservoir training:
Initialize: $\mathbf{W}_{out} = 0$, exponential forgetting factor (λ_{RLS}) is set to a value less than 1 (we use 0.85) and the auto-correlation matrix ρ is initialized as $\rho(0) = \mathbf{I}/\beta$, where $\mathbf{I}$ is unit matrix and β is a small constant.

Repeat: At time step t
Step 1: For each input signal $\mathbf{u}(t)$, the reservoir state $\mathbf{x}(t)$ and network output $\mathbf{y}(t)$ are calculated using Eq. 3 and Eq. 4.

Step 2: Online error $e(t)$ calculated as:
$e(t) \leftarrow \delta(t)$

Step 3: Gain vector $\mathbf{K}(t)$ is updated as:
$\mathbf{K}(t) \leftarrow \frac{\rho(t-1)\mathbf{x}(t)}{\lambda_{RLS} + \mathbf{x}^T(t)\rho(t-1)\mathbf{x}(t)}$

Step 4: Update the auto-correlation matrix $\rho(t)$
$\rho(t) \leftarrow \frac{1}{\lambda_{RLS}}\Big[\rho(t-1) - K(t)\mathbf{x}^T(t)\rho(t-1)\Big]$

Step 5: Update the instantaneous output weights $\mathbf{W}_{out}(t)$
$\mathbf{W}_{out}(t) \leftarrow \mathbf{W}_{out}(t-1) + K(t)e(t)$

Step 6: $t \leftarrow t + 1$

Until: Maximum number of time steps is reached.

As proposed in [15][2] we also implement a generic intrinsic plasticity mechanism based on the Weibull distribution for unsupervised adaptation of the reservoir neuron nonlinearity. This allows the reservoir to homoeostatically maintain a stable firing rate while at the same time prevent unwanted chaotic neural activity. The reservoir neurons and the output neurons are updated using a tanh nonlinear activation function i.e. $f_{sys} = f_{out} = \mathtt{tanh}$.

The actor is designed as a stochastic unit, such that for a one dimensional action setup the output (O_{ac}) is given as:

$$o_{ac}(t) = \epsilon(t) + \sum_{i=1}^{K} w_i(t) u_i(t) \tag{5}$$

where K denotes the number of sensory inputs ($\mathbf{u}(t) = u_1(t), u_2(t), .., u_K(t)$) to the agent being controlled. w_i represent the synaptic weights for the different sensory inputs. $\epsilon(t)$ is the exploration quantity updated at every time step such that the agent should explore the environment more if the expected cummulative future reward v is suboptimal and decrease the exploration as v is maximised. As a result one should expect the exploration to tend towards zero as the agent starts to learn the desired behavior. Using a gaussian white noise σ (zero mean and standard deviation one) bounded by the minimum and maximum limits of the value function (v_{min} and v_{max}), the exploration term is modulated as follows (Ω is constant scale factor):

$$\epsilon(t) = \Omega\sigma(t)\Big[\mathtt{min}\Big[0.5, \mathtt{max}\Big(0, \frac{v_{max} - v(t)}{v_{max} - v_{min}}\Big)\Big]\Big] \tag{6}$$

The actor learns by an online adaptation (Fig. 2 above) of its synaptic weights w_i at each time step modulated by the TD-error $\delta(t)$ from the Critic network (Equation (2)) as follows:

$$\Delta w_i(t) = \alpha\delta(t)u_i(t)\epsilon(t) \tag{7}$$

Where α is the learning rate such that $0 < \alpha < 1$.
Instead of using direct reward to update the actor weights, using TD-error (i.e. error of an internal reward) allows the system to handle even delayed reward control problems. In general once the agent learns the desired behavior, the exploration term ($\epsilon(t)$) should become zero, as a result of which no further weight change (Eq. (7)) occurs and $o_{ac}(t)$ gives the desired action without any noise. The reservoir network being an input driven dynamical system, endows the critic with long temporal memory in contrast to traditional feedforward critic networks (RBF kernels). Specifically in order to solve POMDP scenarios, temporal memory is crucial to propagate the knowledge of previously visited state space (sensory signals) for expected reward in the future. As a result unlike RBF based critics our network can effectively deal with such problems in continuous time.

B. Combinatorial learning with reward modulated Hebbian plasticity

In the previous subsections we provided an overview of the combinatorial learning framework along with the description of the new dynamic reservoir based actor-critic reinforcement learning network. We now elaborate on the parallel combination of the correlation-based learner (ICO) and the reward-based learner (actor-critic) as depicted in Fig. 1. The system works as a dual learner where the individual learning mechanisms run in parallel to guide the behavior of the agent. Both the systems adapt their weights independently while receiving sensory feedback from the agent (system state) in parallel. The final action that drives the agent is calculated as a weighted sum of the individual components. This can be described as follows:

$$o_{com}(t) = \xi_{ico} o_{ico}(t) + \xi_{ac} o_{ac}(t) \tag{8}$$

where, $o_{ico}(t)$ and $o_{ac}(t)$ are the t time step outputs of the input correlation-based learner and the actor-critic learner, respectively. $o_{com}(t)$ represents the t time step combinatorial action. The important parameter here is the weights of the individual components (ξ_{ico} and ξ_{ac}) that govern their degree of influence on the net action of the agent. A simple and straight forward approach [8] is to provide equal contribution ($\xi_{ico} = \xi_{ac} = 0.5$) for controlling the agent. Although this leads to successful solutions, they are sub-optimal. Intuitively for associative learning problems with immediate rewards the ICO system learns quickly as compared to distal reward based goal-directed problems where the ICO learner provides guidance to actor-critic learner. In general depending on the type of problem, the interaction between the two learning systems differs and needs to be taken into account. We solve this problem by introducing a new plasticity rule called reward modulated hebbian plasticity [6] in order to learn the individual synaptic weights. Based on this plasticity rule the ICO and actor-critic RL weights are learnt at each time step as follows :

$$\Delta\xi_{ico}(t) = \eta r(t)(o_{ico}(t) - \bar{o}_{ico}(t))o_{ac}(t), \tag{9}$$

$$\Delta\xi_{ac}(t) = \eta r(t)(o_{ac}(t) - \bar{o}_{ac}(t))o_{ico}(t). \tag{10}$$

Here $r(t)$ is the current time step reward signal received by the agent, while $\bar{o}_{ico}(t)$ and $\bar{o}_{ac}(t)$ denote the low-pass filtered version ($\bar{o}_{ico,ac}(t) = 0.9\bar{o}_{ico,ac}(t-1) + 0.1 o_{ico,ac}(t)$) of the output from the ICO learner and the actor-critic learner, respectively. The plasticity model used here is based on the assumption that the net policy performance (agents behavior) is influenced by a single global neuromodulatory signal. The learning rule measures correlations between the reward signal and the deviations of the ICO and actor-critic learner outputs from their mean values and accordingly adjusts the respective weights. In order to prevent uncontrolled divergence in the learnt weights (ξ_{ico} and ξ_{ac}), synaptic normalization is introduced by dividing the individual weights by the total sum of weights. This ensures that the weights always add up to one and $0 < \xi_{ico}, \xi_{ac} < 1$. In general this plasticty rule occurs on a slow time scale which is governed by the learning rate parameter η. Typically η is set much less compared to the learning rate of the two individual learning systems (ICO and actor-critic).

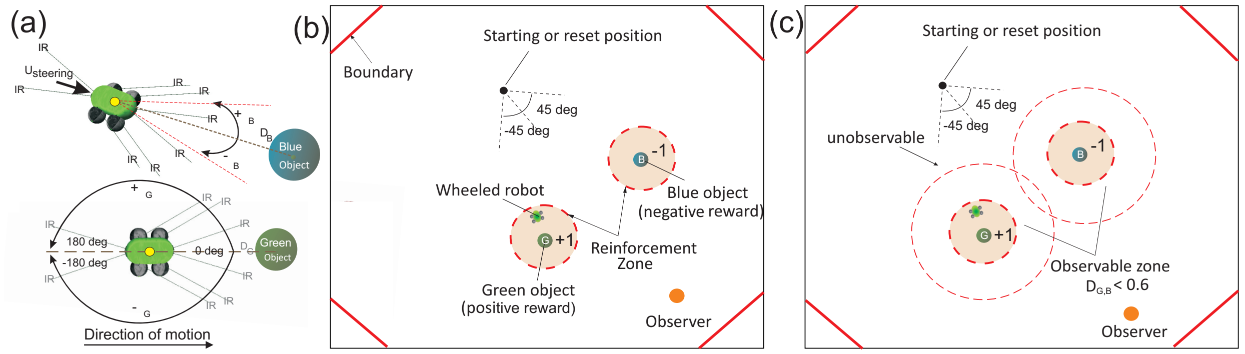


Fig. 3. Simulated mobile robot system for goal-directed behavior task. (a) (Top) The mobile robot NIMM4 with different types of sensors. The relative orientation sensor ϕ is used as state information for the robot. (Bottom) Variation of the relative orientation ϕ_G to the green goal. (b) Environmental setup for the fully observable case. The robot continuously senses its relative orientation to both the green and blue objects. Only whithin the reinforcement zone (shaded circle) the robot receives positive reward when near the green goal and negative reward when near the blue goal. (c) Environmental setup for the partially observable case. The robot can sense its relative orientation to the goals only when within the observable zone (outer dotted circles). Reinforcement is received similar to the fully observable case. Here the orange object represents an external observer.

III. Experiments and Results

In order to test the performance of the combinatorial learning framework with a reservoir critic and reward modulated Hebbian plasticity, we employ a goal-directed behavior control task using a simulated wheeled robot system (Fig. 3 (a)). The task is to let the wheeled robot NIMM4 learn to steer itself towards a desired goal (green ball, Figs. 3(b) and (c)) within a given time. As the robot approaches the desired goal, it receives positive reinforcement. Additionally an undesired goal (blue spherical ball) with negative reinforcement was also placed within the same arena. NIMM4 is provided with two relative orientation sensors (ϕ_G - green ball, ϕ_B - blue ball) that can measure angle of deviations from the two goals. They can take values in the interval $[-180^o, 180^o]$ with the $\phi_{G,B} = 0^o$ when the respective goal is directly in front of the robot. In addition NIMM4 also consists of two relative position sensors ($D_{G,B}$) that can calculate it's relative distance to a goal in the interval [0,1] with the respective sensor reading tending to zero, as the robot gets closer to a goal. The task is further divided into fully and partially obervable scenarios. In the first case, the robot can continuously sense its angle of deviation to the two goals with $\phi_{G,B}$ always active. For the later case, the robot cannot sense direction to either of the goals ($\phi_{G,B}$ inactive) untill it reaches the half way distance to either of the goals i.e. $D_{G,B} < 0.6$. In both the cases when the robot gets very close to either of the goals, within a distance of ($D_{G,B} = 0.2$) it receives a positive or negative reward. Within this boundary for the green goal it receives a continuous reward of +1 at every time step and a continuous reward of -1 in case of the blue goal, respectively. This distance is also used as the zone of reflex to trigger a reflex signal for the ICO learner. It is important to note that only the relative orientation sensory data is used as state input for both the ICO learner and the actor-critic learner. Furthermore as $\phi_{G,B}$ signals overlap with each other (i.e., the robot simultaneously senses its relative orientation to both the goals in the whole arena). NIMM4 is also supplied with eight infra-red sensors that are used only to reset it to the starting location if it hits a boundary before reaching either of the

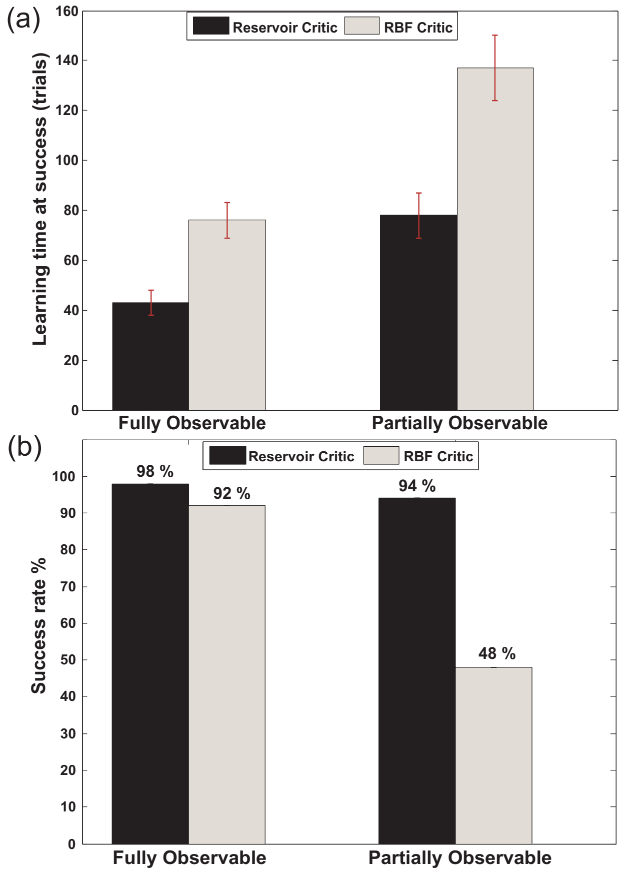


Fig. 4. Performance comparison between a reservoir based critic and RBF based critic for the fully observable and partialy observable cases (ICO and actor components remained the same). (a) Average learning time (trials) needed to succesfully complete the task, calculated over 50 experiments (error bars indicate standard deviation for 95 % confidence interval). (b) Success rate in percentage. Here "success" indicates the robots ability to correctly navigate to the green goal.

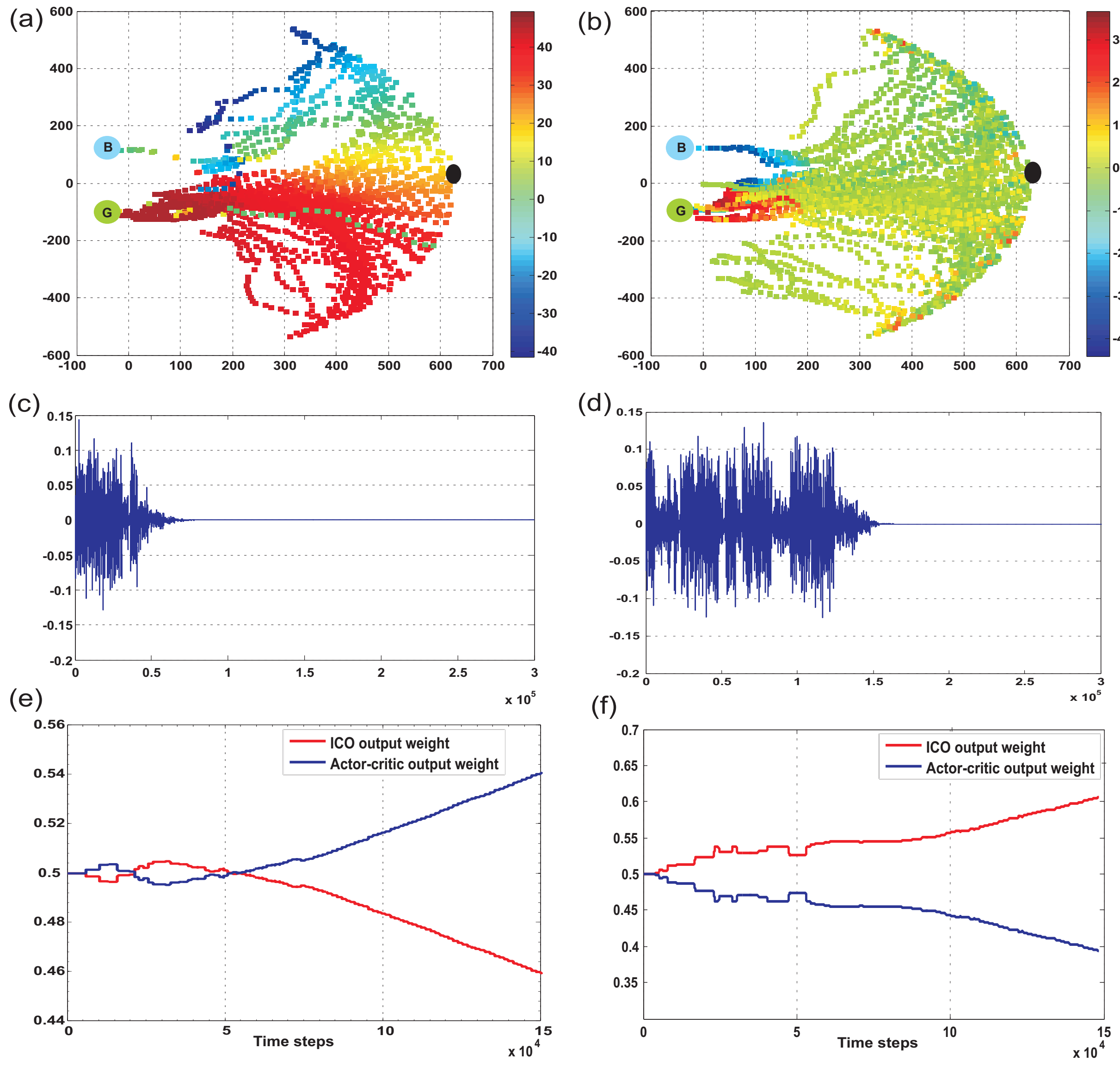


Fig. 5. (a) Estimation of the value function $v(t)$ using reservoir based critic. The $v(t)$ estimate is plotted with respect to local co-ordinates of the robot and an observer located directly opposite to the robot starting position. Colormap indicates the changing $v(t)$ values. The black ball indicates the starting position of the robot with random orientation and the curvature of the plot is resultant of the shape of view from the observer. (b) Estimation of value function $v(t)$ for the same task using the static RBF based critic. (c) Convergence of exploration term $\epsilon(t)$ using reservoir critic. (d) Convergence of exploration term using RBF critic. (e) Adaptation of ICO weights ξ_{ico} and actor-critic weights ξ_{ac} using the RMHP rule for Combined learner with reservoir critic. (f) Adaptation of ICO weights ξ_{ico} and actor-critic weights ξ_{ac} using the RMHP rule for Combined learner with RBF critic. The continuous change in the learned weights even after successful learning of task is due to the Hebbian nature of the adaptation rule. This can be easily controlled by introducing additional synaptic scaling mechanism or in this case stop the weight updation once exploration ($\epsilon(t)$) becomes zero. All the plots were generated for the same goal-directed behavior task of reaching the green goal. The plots indicate the best performance case for each setup.

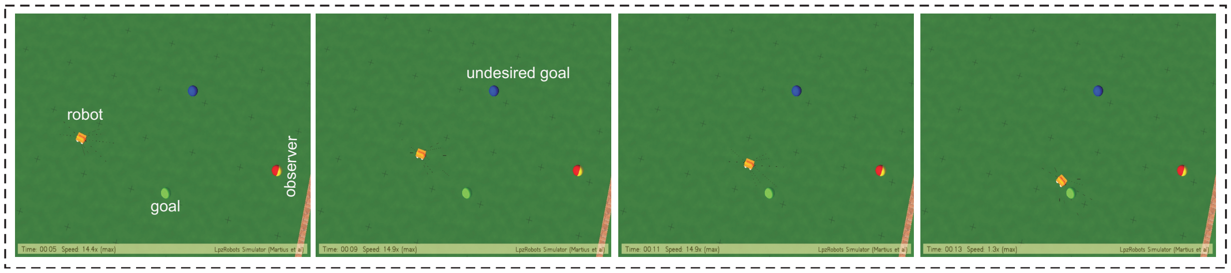


Fig. 6. Simulation screenshots showing the actual behavior of the robot after succesfully learning the task. Upon learning, it continuously steers towards the green goal and avoids movements towards the blue ball. For video of a complete learning sequence, please visit http://www.manoonpong.com/rcrl/rcrl.wmv .

goals. Keeping the ICO learner fixed for the combinatorial setup, we tested both the scenarios (Figs. 3(b) and (c)) for a reservoir based critic and a feedforward RBF critic. The combinatorial learning mechanism learns to steer the robot towards the desired goal (green object). Without control, the robot randomly moved around. The robot always starts from the same location, however with random orientation. 50 runs were carried out with each setup for both fully observable and partially-observable scenarios. Each run consisted of a maximum of 200 trials (robot resets). The robot was reset if it reached either of the goals or if it hit a boundary wall or if the maximum simulation time of 15s was reached.

ICO learning was setup as follows: $\phi_{G,B}$ were used as predictive signals. Two independent reflex signals were configured with one for blue ball and the other for the green ball. The reflex signal was designed to elicit a turn towards a ball once the robot comes close enough to it (inside the dotted circle in Figs. 3(b) and (c)). Irrespective of the kind of goal (desired or undesired) the reflex signal drives the robot towards it with a turn proportional to the deviations defined by $\phi_{G,B}$ i.e large deviations cause sharper turns. The green and the blue ball were placed such that there was no overlap between the reflex areas, hence only one reflex signal got triggered at a time. In other words, the goal of the ICO learner is simply to learn to drive towards a goal location without any knowledge of their worth (positive or negative reward).

The actor-critic learner was setup as follows: The inputs to the critic and actor networks (Fig. 2) consisted of the two relative orientation sensor data ϕ_G and ϕ_B. The reservoir network for the critic consisted of $N = 100$ neurons and one ouput neuron that estimates the value function $v(t)$ (Eq. (1)). Reservoir input weights W_{in} were drawn from an uniform distribution $[-0.5, 0.5]$ while the reservoir recurrent weights W_{sys} were drawn from the uniform distribution $[-1, 1]$. W_{sys} was subsequently scaled to a spectral radius of 0.9 with only 10% internal connectivity. The reward signal $r(t)$ (Eq. (2)) was set to +1 when the robot comes close to the green ball and to -1 when it comes close to the blue ball. A RBF feedforward network was used for comparison with the reservoir based critic. The RBF critic size was varied from 16 to 100 hidden neurons. All other combinatorial network parameters are summarised in Table 1.

The performance of the reservoir based critic as compared to the RBF critic (keeping all other components of the combinatorial learning framework the same) is compared in Fig. 4 with respect to the fully and the partially observable scenarios of the same task. As observed from Fig. 4(b), the reservoir based critic clearly outperforms the RBF critic. Moreover the difference in performance is highly significant in the POMDP scenario, where the reservoir network outperforms the RBF critic by a success rate greater than 50%. Temporal memory of incoming agent state information available to the reservoir critic is crucial for solving complex non-markovian problems, as compared to memoryless feedforward critic networks. Furthermore although both the implementations have almost similar success rate for the fully observable case, the reservoir based system converges to a solution (learned behavior of driving the robot to the green goal) faster (less than 50 trials), as observed in Fig. 4(a). However, expectedly the POMDP scenario takes longer time to learn the correct behavior, owing to the reduction in the total sensory information available to the system. Upon successfully learning the task the weights of the actor (Eq. 7) converge such that the robot gets pulled towards the desired green goal. It should be noted that although linear actors (Eq. 5) were used in this setup, the POMDP scenario is effectively solved due to the inherent trace of previous inputs in the reservoir critic. In contrast the memoryless RBF critic system works on chance and hence learns the POMDP task with less than 50% success rate.

In Figs. 5 (a) and (b) we compare the performance of the reservoir based critic with a RBF critic network in terms of the value function estimation curves for the same goal-directed behavior task (i.e. the fully observable task). It is clearly observed that the reservoir critic successfully enables the mobile robot to learn to drive towards the green goal while avoiding the blue goal. Furthermore unlike the RBF critic (Fig. 5(b)), the value function curve in Fig. 5(a) displays a strong gradient of the estimated value of $v(t)$ with high positive values towards the correct goal (green object). In contrast the memory less RBF critic estimates $v(t)$ to values closer to zero in most locations except for regions within the zone of reward. As a result our modified critic learns the task faster as indicated by the fast convergence of the exploration term $\epsilon(t)$ in Figs. 5 (c) and (d). In Figs. 5(e) and (f) we plot the development of the ICO ξ_{ico} and actor-critic weights ξ_{ac} via the RMHP learning rule (Eqs. (9) and (10)). In case of the reservoir critic, the actor-critic learner component is seen to dominate over the ICO learner. In contrast when the RBF critic was used for the same task, the learnt behavior is dominated by the ICO component. This can be explained in terms of the memory of sensory state present in the reservoir network that successfully guides the agents behavior in contrast to the

TABLE I.

The List of combinatorial network parameters	
Reservoir critic size (neurons)	100
RBF critic size (neurons)	16 - 100
Reservoir leak rate (λ)	0.3
RLS learning constant (β)	10^{-2}
Discount factor (γ)	0.95
Scale factor (Ω)	5
Maximum value (v_{max})	50.0
Minimum value (v_{min})	-50.0
Neuron non-linearity (f_{sys},f_{out})	tanh
RLS learning rate (λ_{RLS})	0.85
Actor learning rate (α)	0.001
RMHP learning rate (η)	0.0005

memoryless RBF network. In general the weight adaptation should occur in a task dependent manner. The actual behavior of the robot NIMM4 after succesfully learning the task of navigating towards the green goal, is depicted via screenshots of the simulation in Fig. 6.

IV. Conclusion

In this work we have successfully extended the neural combinatorial learning framework (CLF) using a reservoir network based adaptive critic, while using a stochastic linear actor unit and a basic implementation of input correlation learning. The resultant network effectively solves goal directed behavioral problems and outperforms the CLF with traditional radial basis function (feed-forward network) based critics both in terms of rate of success and the overall learning time. Furthermore due to the inherent temporal memory of reservoir networks, our modified critic enables the CLF to solve partially observable scenarios. In addition we implement a new biologically plausible reward modulated Hebbian plasticity rule which enables the CLF to learn the degree of influence of the ICO learner as compared to the actor-critic learner. This allows automatic weight adaptation between the two components working on a very slow time scale. As a future direction we plan to extend the linear actor units to reservoir based nonlinear actors which would work in conjunction with the reservoir critic. This would enable the controlled agent with memory capabilities in the action domain and thereby solve more complex goal-directed behavioral tasks like delayed reward maze navigation. Moreover behavioral analysis of the RMHP rule for evaluating the stability of learning against changing metaparameters (E.g. individual learning rates) with task independent evaluation measures will be carried out as further extension of our current work.

Acknowledgment

This research was supported by the Emmy Noether Program (DFG, MA4464/3-1), the Federal Ministry of Education and Research (BMBF) by a grant to the Bernstein Center for Computational Neuroscience II Göttingen (01GQ1005A, project D1), the Strategic Japanese-German Cooperative Program on Computational Neuroscience (JST-DFG, WO388/11-1), European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 270273 (Xperience).

References

[1] Doya, K. (2000) Reinforcement Learning In Continuous Time and Space. Neural Computation, 12, 219-245.

[2] Dasgupta, S., Wörgötter, F., and Manoonpong, P. (2013) Information Dynamics based Self-adaptive Reservoir for Delay Temporal Memory Tasks. Evolving Systems, doi: 10.1007/s12530-013-9080-y.

[3] Frémaux, N., Sprekeler, H., Gerstner, W. (2013), Reinforcement Learning Using a Continuous Time Actor-Critic Framework with Spiking Neurons. PLoS Comput Biol 9(4): e1003024. doi:10.1371/journal.pcbi.1003024.

[4] Jaeger, H., and Haas., H. (2004): Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. Science, 304(5667), 78-80.

[5] Maass, W., Natschlger, T., and Markram, H. (2002), Real-time computing without stable states: a new framework for neural computation based on perturbations. Neural Computation. 14 (11): 253160.

[6] Legenstein, R., Chase, S.M., Schwartz, A.B., and Maass, W. (2010) A reward-modulated hebbian learning rule can explain experimentally observed network reorganization in a brain control task. J Neurosci 30:84008410.

[7] Manoonpong, P., Wörgötter, F., and Morimoto, J. (2010) Extraction of Reward-Related Feature Space Using Correlation-Based and Reward-Based Learning Methods. In Proc. 17th International Conference on Neural Information Processing, Sydney, Australia, November 22-25 (ICONIP'10),Part I, LNCS 6443, pp. 414-421.

[8] Manoonpong, P., Kolodziejski, C., Wörgötter, F., and Morimoto J. (2013) Combining Correlation-Based and Reward-Based Learning in Neural Control for Policy Improvement. Advances in Complex Systems, doi: 10.1142/S021952591350015X.

[9] Morimoto, J., and Kenji, Doya. (1998) Reinforcement learning of dynamic motor sequence: Learning to stand up. In Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 3. IEEE.

[10] Oubbati, M., Kchele, M., Koprinkova-Hristova, P., and Palm, G. (2011), Anticipating Rewards in Continuous Time and Space with Echo State Networks and Actor-Critic Design. In Proc. 19th European Symposium on Artificial Neural Networks (ESANN).

[11] Koprinkova-Hristova, P., Oubbati, M., and Palm, G. (2010). Adaptive critic design with echo state network. In Proc. IEEE International Conference on Systems, Man, and Cybernetics, 1010-1015.

[12] Pavlov, I., Conditioned reflexes (Oxford University Press, Oxford, UK, 1927)

[13] Porr, B., and Wörgötter, F. (2006), Strongly improved stability and faster convergence of temporal sequence learning by utilising input correlations only. Neural computation 18, 1380-1412.

[14] Skinner, B., The Behavior of Organisms: An Experimental Analysis (Appleton Century Croft, New York,1938)

[15] Triesch, J. (2007), Synergies between Intrinsic and Synaptic Plasticity Mechanisms. Neural Computation 4, 885-909.

[16] Wörgötter, F., and Porr, B. (2004) Temporal sequence learning, prediction and control - a review of different models and their relation to biological mechanism. Neural Computation. 17, 245-319.

Advances in Complex Systems
Vol. 16 (2013) 1350015 (38 pages)
© World Scientific Publishing Company
DOI: 10.1142/S021952591350015X

COMBINING CORRELATION-BASED AND REWARD-BASED LEARNING IN NEURAL CONTROL FOR POLICY IMPROVEMENT

PORAMATE MANOONPONG[*,†,‡], CHRISTOPH KOLODZIEJSKI[*,§]
FLORENTIN WÖRGÖTTER[*,¶] and JUN MORIMOTO[*,†,||]

[*]*Bernstein Center for Computational Neuroscience,
The Third Institute of Physics,
University of Göttingen, Göttingen 37077, Germany*

[†]*ATR Computational Neuroscience Laboratories,
2-2-2 Hikaridai Seika-cho,
Soraku-gun, Kyoto 619-0288, Japan*
[‡]*poramate@physik3.gwdg.de*
[§]*kolo@physik3.gwdg.de*
[¶]*worgott@physik3.gwdg.de*
[||]*xmorimo@atr.jp*

Received 21 February 2012
Revised 6 February 2013
Accepted 20 February 2013
Published 24 April 2013

Classical conditioning (conventionally modeled as correlation-based learning) and operant conditioning (conventionally modeled as reinforcement learning or reward-based learning) have been found in biological systems. Evidence shows that these two mechanisms strongly involve learning about associations. Based on these biological findings, we propose a new learning model to achieve successful control policies for artificial systems. This model combines correlation-based learning using input correlation learning (ICO learning) and reward-based learning using continuous actor–critic reinforcement learning (RL), thereby working as a dual learner system. The model performance is evaluated by simulations of a cart-pole system as a dynamic motion control problem and a mobile robot system as a goal-directed behavior control problem. Results show that the model can strongly improve pole balancing control policy, i.e., it allows the controller to learn stabilizing the pole in the largest domain of initial conditions compared to the results obtained when using a single learning mechanism. This model can also find a successful control policy for goal-directed behavior, i.e., the robot can effectively learn to approach a given goal compared to its individual components. Thus, the study pursued here sharpens our understanding of how two different learning mechanisms can be combined and complement each other for solving complex tasks.

Keywords: Classical conditioning; operant conditioning; associative learning; reinforcement learning; pole balancing; goal-directed behavior.

1. Introduction

In biological systems, two classes of conditioning for associative learning are known [5]. One is classical conditioning [50] involving presentations of a conditional stimulus (CS) along with a significant or unconditional stimulus (US). The US generally drives an unconditional response (UCR), usually a reflex (e.g., salivation in dogs when they encounter food). Once the US and CS become associated, animals begin to perform a behavioral response to the CS rather than the US where this response is called a conditional response (CR). This modification basically happens only if the CS is a predictor for the US [56]. Thus, the CS normally precedes the US ([50, 70], but see Ref. [5] for detailed clarification). Another type of conditioning is operant or instrumental conditioning [59, 63]. It mainly involves a reinforcer (i.e., a US) associated with behavior modification instead of another stimulus. The probability of a specific behavior is increased or decreased through positive or negative reinforcement at each time that the reinforcement is generated.

Although these conditioning or learning mechanisms are different from each other, a number of studies on animal leaning suggest that they may act in combination [5, 15, 35, 47, 55], rather than separately or alternatively, to obtain an appropriate behavior. Experiments that have supported this idea were presented in, e.g., Refs. [11, 39, 68]. Williams and Williams [68] observed a pigeon pecking at an illuminated key in a Skinner box. The results suggest that the desired key-pecking behavior CR may be shaped by not only operant conditioning[a] but also by classical conditioning; since imposing an omission schedule on the key-light, key-peck association did little to revoke the conditional pecking response. Hence, it seems that the existing occasional pairing of the key-light CS with the food US was adequate to drive the pecking behavior (CR), which thus emerged from classical conditioning. Lovibond [39] performed experiments in rabbits by providing separately trained conditional stimuli during reinforced operant responding. His results showed that the strength of an operant response can be influenced by simultaneously presenting a classically CS. Brembs and Heisenberg [11] conducted experiments in the fruit flies (Drosophila). Their results showed that there is a situation where both operant and classical predictors play their roles at the same time, such that the situation can be more easily learned than in the separate case.

In animal training, evidence also reveals that many animals including rodents, dogs, pigeons, dolphins, seals, and whales, can effectively learn to do some sophisticated tasks when they are trained using a combination of these mechanisms [25]. For instance, marine animal trainers use a whistle as predictive information to "tell" their animals that a reward (e.g., food) is forthcoming. Thus, marine animals learn to associate the sound of whistle and food (i.e., learning via classical conditioning). When the animals perform a desired behavior (e.g., come, jump, flip, etc.), they

[a]In this situation, the animal was induced to respond to the key in association with a reward (i.e., food). This procedure is also called autoshaping.

first hear the sound indicating that they have performed appropriately and then they receive food (i.e., learning via operant conditioning). After several repetitions, the animals will perform a certain behavior as soon as they hear the sound where they expect to receive food afterwards.

Classical conditioning is often modeled as a form of correlation-based (differential Hebbian) learning [32, 37, 52, 70] in computational neuroscience. This approach uses the correlations between external stimuli (i.e., the US and CS) for synaptic plasticity leading to an anticipatory action (see Sec. 2 for more details). Operant conditioning is often modeled as reward-based learning or reinforcement learning (RL, e.g., temporal difference (TD)-learning [7, 62, 67]) in computer science. This approach uses predefined rewards and punishments in the environment as evaluation allowing an agent to maximize or optimize its own expected cumulative future reward (or expected return). As a consequence, this leads to a corresponding behavior (see Sec. 3 for more details).

These two conditioning concepts or learning mechanisms have been widely applied to artificial agents (robots) for solving various tasks including the generation of self-organizing behavior and autonomous systems [8]. Generally, much research has *separately* used them to enable agents to learn solving their tasks [7, 10, 20, 38, 40, 42, 51, 53, 62]. In this study, we point out that these two learning frameworks can complement each other leading to policy improvement. Correlation-based learning can quickly find a correlation between a state and an unwanted condition (i.e., reflex or failure recognized by an immediate reflex signal), but cannot evaluate whether a given state or weight change predicts something "good" or "bad" which will happen many steps away in the future. Consequently, it cannot properly learn solving some difficult tasks (e.g., delayed reward tasks) and cannot explicitly derive a goal-directed policy. On the other hand, reward-based learning can derive a policy according to the (delayed) reward signal but using it without any prior knowledge (predefined control parameters), environment or system models, or appropriate guidance generally takes many learning trials to improve the control performance. Therefore, we combine correlation-based learning (using input correlation learning (ICO learning) [52]) and RL (using continuous actor–critic RL [19]) in parallel to let ICO learning extract important features directly used to guide the learning strategy of continuous actor–critic RL. If we can extract important or proper features for the task, a model of the policy can be simple and the policy can be easily improved.

To investigate this hypothesis, to show how these two biologically-inspired learning mechanisms can be combined as a neural learning system, and to present its performance, we chose pole balancing and goal-directed behavior control problems as two different case studies or testbeds. Generally, we are not interested in solving these two tasks *per se*. Instead, we would like to show that the proposed combination can solve model-free control problems and is not limited to a specific task. Additionally, we would like to suggest that this combination can be an advantageous but simple way (i.e., only combining them in parallel without modifying their learning mechanisms) to solve (dynamic) sensorimotor control problems with continuous signals. Through the performed experiments we hope that this model may help to better understand interactions between the two learning mechanisms. To a certain extent, the model might be related to neural learning in biological systems and it may provide a computationally oriented perspective on animal learning. Finally, we would like to emphasize that this combinatorial learning framework suggests how a prior knowledge can be provided to RL and how RL can be guided and shaped for policy improvement. To our knowledge, this kind of combinatorial learning method (which is simple, partially related to neural learning mechanisms in the brain — see the Discussion and Conclusion section below — and leads to policy improve) has not been investigated and presented so far.

This paper is organized as follows. In Sec. 2 we present the neural circuit of ICO learning while the neural circuit of continuous actor–critic RL is given in Sec. 3. In Sec. 4, we introduce our learning model which combines both learning mechanisms inspired by biological findings. This model will lead to policy improvement. In Sec. 5, we demonstrate its performance using the pole balancing and goal-directed behavior control problems and provide a comparison of different learning mechanisms. This paper finishes in Sec. 6 with discussion and conclusions.

2. Correlation-Based Learning

For correlation-based learning, we used ICO rule [52] (see Fig. 1) since this learning rule allows implementation of fast and stable learning and it has been also successfully applied to real robots for obtaining adaptive behavior [40, 42, 53].

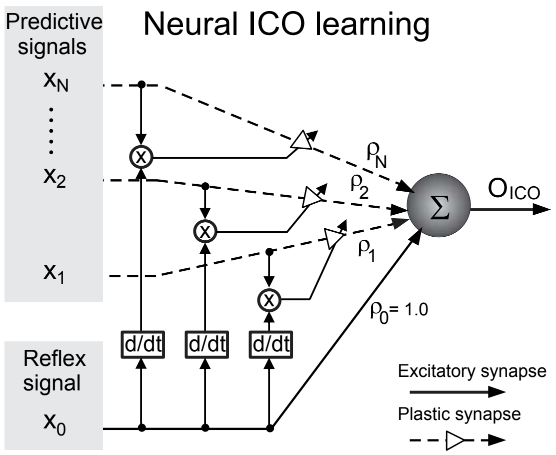


Fig. 1. Neural circuit of ICO learning for the multiple plastic synapses of predictive inputs. The learning rule is derived from differential Hebbian learning. Here, the output neuron (or learner neuron) was modeled as a simple linear neuron (see text for details). It generates a continuous signal for controlling a system.

ICO learning is a form of online unsupervised learning where its rule for synaptic adaptation is based on the cross-correlation of the two types of input signals: Multiple predictive signals (here considered as CS) which are earlier occurring stimuli and a single reflex signal (here considered as a US) which arrives later with certain delays and drives an unwanted response (or reflex). The learning goal of ICO learning is to use a predictive signal ((observable) state of the system) to predict the occurrence of a reflex signal (some exogenous immediate feedback, e.g., reaching a failure state), thereby allowing an agent to react earlier. In other words, this learning mechanism enables the agent to learn to perform an anticipatory action to avoid the reflex. For example, heat radiation (predictive signal) precedes a pain signal (reflex signal) when touching a hot surface. Thus, we learn an anticipatory action to avoid the late unwanted stimulus (i.e., avoiding to touch the hot surface).

Normally, the synaptic adaptation of ICO learning changes through heterosynaptic interactions [27] as a consequence of the order of the arriving inputs. If the predictive inputs are followed by the reflex input, the plastic synapses of the predictive inputs get strengthened but they get weakened if the order is reversed. Hence, this form of plasticity depends on the timing of correlated neural signals. Formally, we have

$$O_{\mathrm{ICO}}(t) = \rho_0 x_0(t) + \sum_{k=1}^{N} \rho_k(t) x_k(t) \tag{1}$$

as the output neuron (O_{ICO}) driven by a linear combination of the reflex input (x_0) and the multiple predictive inputs (x_k). N denotes the number of predictive inputs. ρ_0 is the synaptic strength of the reflex input. This synaptic strength is set to a positive value, e.g., 1.0, and remains unchanged, like an innate reflex. During learning, the plastic synapses (ρ_k) get changed by differential Hebbian learning [32, 37] using the cross-correlation between both inputs (i.e., x_0 and x_k). This is expressed as:

$$\frac{d\rho_k(t)}{dt} = \mu x_k(t) \frac{dx_0(t)}{dt}, \quad k = 1, \ldots, N. \tag{2}$$

μ is the learning rate which defines how fast a system can learn. It is generally set to a value smaller than 1.0. This learning mechanism leads to weight stabilization as soon as $x_0 = 0$ [52], meaning that the reflex has been successfully avoided. As a result, we obtain behavioral and synaptic stability at the same time without any additional weight-control mechanisms.

Due to the learning rule, ICO learning can be considered as a model-free method since its does not require a system or environment model. However, one should note that ICO learning requires the proper design of a reflex into the system from the beginning. This means that we have to set up a feedback system which has a desired state and an error signal ($x_0 \to 0$) which drives learning. If the tasks become more complex where a reflex cannot be properly designed or no correlation between a reflex signal and a predictive signal exists at all, ICO learning will fail.

3. Reward-Based Learning

For reward-based learning, we used continuous actor–critic RL [19] (see Fig. 2) since it is capable of generating (multidimensional) continuous actions thus providing a smooth control performance. It is also practical for continuous state-action problems [19], like dynamic motion control [20, 46]. In addition, it is based on a biological learning model [70] where its learning rule for synaptic adaptation considers an association between stimuli and/or actions with the reinforcement that an agent receives. Formally, the reinforcement is a reward or a punishment which is "evaluative feedback" defined by the designers of a system. Thus, this kind of

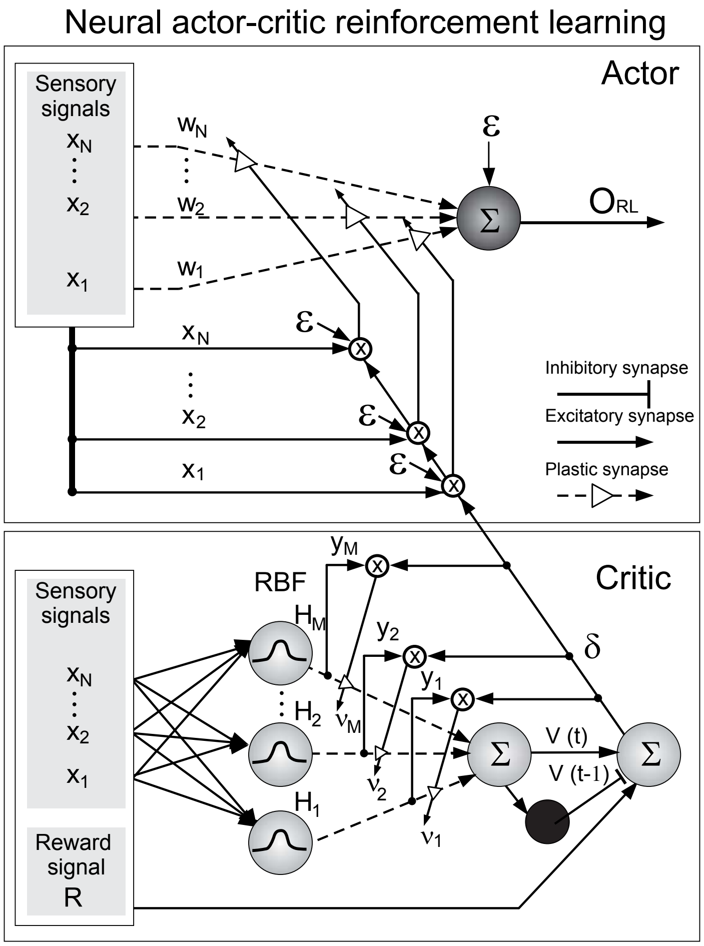


Fig. 2. Neural circuit of continuous actor–critic RL. The learning mechanisms of the actor and critic are based on TD learning. The actor component was modeled as a stochastic neural network while the critic unit was modeled as a radial basis function (RBF) neural network (see text for details). Note that in this framework, the actor provides a continuous output signal for controlling a system.

learning mechanism is minimally supervised because an agent is not told explicitly what actions to take in a particular situation. Rather it must work this out for itself on the basis of the reinforcement.

Continuous actor–critic RL is divided into two sub-mechanisms: The learning of an action function (actor) and the learning of an evaluation function (critic). The action part is the controller of an agent. In this study, it was designed as a stochastic unit proposed in Ref. [24]. If we consider one-dimensional output, its output (O_{RL}) is specified by:

$$O_{\mathrm{RL}}(t) = \varepsilon(t) + \sum_{k=1}^{N} w_k(t) x_k(t), \tag{3}$$

where N denotes the number of sensory inputs (x_k) which, here, are comparable to the predictive inputs of ICO learning. ε is an exploration term. According to Ref. [19], it is varied based on a modulation scheme[b] given by:

$$\varepsilon(t) = \xi\sigma(t) \cdot \min\left[1, \max\left[0, \frac{V_{\max} - V(\mathbf{x}(t))}{V_{\max} - V_{\min}}\right]\right]. \tag{4}$$

σ is the Gaussian distributed noise with zero mean and standard deviation of one. V is a value function (see its equation below) that estimates the expected cumulative future reward or the expected return where the reward is used to estimate how good it is for an agent to be in a given state. $V_{\max}$ and $V_{\min}$ are the maximal and minimal values of V. This way, the exploration is large if V is close to $V_{\min}$. On the other hand, the exploration is small (close to zero) if it is close to $V_{\max}$ meaning that learning shows good prediction or the performance is improved. ξ is an additional scale factor. It is introduced in order to be able to amplify the exploration level.

The stochastic unit is related to two biological learning concepts, called behavior oscillation [26] and successive approximation [59] (see also Ref. [24] for more details). During learning, the synaptic weights (w_k) of the actor change over time. They are basically changed by a stochastic RL algorithm [24]. Instead of using the error of a direct reward, which is one of the learning parameters and originally used in the stochastic RL algorithm, here we used the TD error [7, 19] (i.e., the error of an internal reward [7]). By doing so, delayed reward control problems can also be solved [7, 19]. The equation of the weight adaptation is described by:

$$\frac{dw_k(t)}{dt} = \alpha\delta(t)x_k(t)\varepsilon(t), \quad k = 1, \ldots, N, \tag{5}$$

where α is the learning rate and generally set to a value smaller than 1.0. δ is an approximation to the TD error in continuous time described as an internal reinforcement signal provided by the critic (see below).

[b]The scheme follows the intuition that an agent should explore a lot if its expected cumulative future reward V is small (close to $V_{\min}$). This means that it has a poor control policy. On the other hand, it should exploit or follow the control policy if V is close to $V_{\max}$. However, this normally works if $V_{\min}$ and $V_{\max}$ could be estimated.

For the critic network, according to Ref. [45] we used a radial basis function (RBF) neural network as a function approximator which attempts to construct the approximation of the value function V. It is governed by:

$$V(\mathbf{x}(t)) = \sum_{j=1}^{M} v_j(t) y_j(\mathbf{x}(t)), \tag{6}$$

where y_j are the outputs from the normalized Gaussian basis functions given by:

$$y_j(\mathbf{x}(t)) = \frac{a_j(\mathbf{x}(t))}{\sum_{l=1}^{M} a_l(\mathbf{x}(t))}, \quad a_j(\mathbf{x}(t)) = e^{-\|\mathbf{s}_j^T(\mathbf{x}(t)-\mathbf{c}_j)\|^2}. \tag{7}$$

The vectors $\mathbf{c}_j$ and $\mathbf{x}$ define the center and the input feature, respectively. $\mathbf{s}_j$ is the diagonal matrix of the inverse covariance of the RBF neural network. M is the number of hidden neurons. According to Ref. [19], v_j are synaptic weights which are updated by:

$$\frac{dv_j(t)}{dt} = \lambda\delta(t) y_j(\mathbf{x}(t)), \quad j = 1, \ldots, M, \tag{8}$$

where λ is the learning rate. It is generally set to a value smaller than 1.0. According to Ref. [19], the TD-error δ is basically computed from the prediction as follows:

$$\delta(t) = R(t) - \frac{1}{\tau} V(\mathbf{x}(t)) + \dot{V}(\mathbf{x}(t)), \tag{9}$$

where R is an external reinforcement signal provided by designers. τ is the time constant of a discount factor. V is the value function [see Eq. (6)] and $\dot{V}$ is its derivative with respect to time. Note that using the Euler discretization, the TD error in continuous time is compatible to the conventional TD error [62]: $\delta(t) = R(t) + \gamma V(\mathbf{x}(t)) - V(\mathbf{x}(t-1))$ where $\gamma = 1 - \frac{\Delta t}{\tau}$ is the discount factor and Δt is the time step of the Euler differentiation.

4. Combining Correlation-Based and Reward-Based Learning

In the previous sections we have presented ICO learning and continuous actor–critic RL. It is known that ICO learning can quickly learn a correlation between a failure state recognized by an immediate reflex signal and a failure avoidance behavior (or also called reflex avoidance behavior) controlled by predictive signals [52]. However, it cannot evaluate whether a given state or weight change predicts something “good” or “bad” which will happen many steps away in the future. As a consequence, this makes it difficult for the controller to properly learn solving some difficult tasks (see Sec. 5). On the other hand, continuous actor–critic RL can make predictions through its evaluation process such that it can solve the tasks but in general it is slower than ICO learning (see Sec. 5). In addition, due to its stochastic process it requires several learning repetitions to ensure that a successful control policy has been achieved.

Thus, in this section, we introduce a combinatorial learning model. It makes use of the advantage of each learning mechanism, resulting in an appropriate acquisition

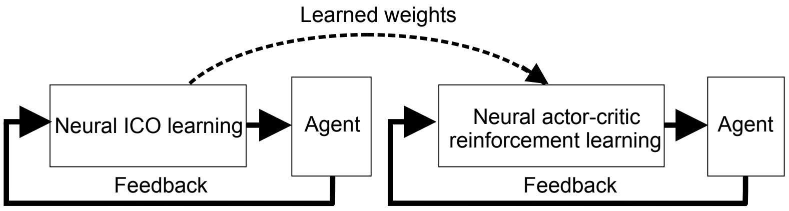


Fig. 3. Sequential combination model. ICO learning first learns to find a solution controlling a system without any prior knowledge. Afterwards the learned weights from ICO learning are provided to continuous actor–critic RL for initialization. Finally, continuous actor–critic RL serves as an add-on learning process to enhance the performance of a controller (see Ref. [43] for more details).

of the control policy that outperforms either ICO learning or continuous actor–critic RL alone (see Sec. 5). Basically, there are two ways of combining ICO learning and continuous actor–critic RL: sequential or parallel.

Sequential combination (see Fig. 3), which we investigated previously [43], is achieved by initially using ICO learning to extract reward-related features for continuous actor–critic RL. Afterwards continuous actor–critic RL uses the extracted features as priors (i.e., initial control parameters) to improve the control policy of the system. However, the drawback of this learning scheme is that it is technically inconvenient since we need to let the ICO learning mechanism learn the whole feature space first such that reward-related features are properly extracted.

In contrast, the parallel combination, proposed in this study and later called here combinatorial learning (see Fig. 4), is technically more convenient since it allows these two learning mechanisms to simultaneously learn, thereby working as a dual learner system. By doing so, they receive sensory feedback from the agent in parallel and adapt their weights accordingly. Their output signal contributes equally to the control of the agent. Thus, the final output (O_{COM}) is described as:

$$O_{\mathrm{COM}}(t) = \zeta \cdot (O_{\mathrm{ICO}}(t) + O_{\mathrm{RL}}(t)), \tag{10}$$

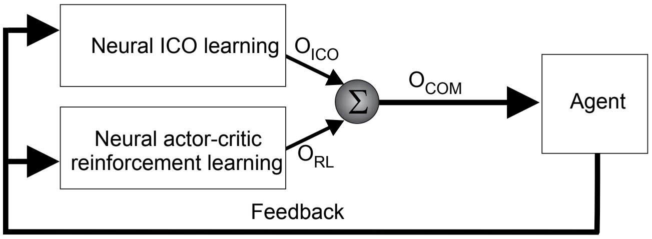


Fig. 4. Combinatorial learning model. It combines ICO learning and continuous actor–critic RL in a parallel manner for controlling an agent. In this learning scheme, each learning mechanism develops its weights independently, but they are coupled by sensory feedback. This way, they basically coadapt the control parameters (i.e., weights) leading to the improvement of the control policy.

where O_{ICO} and O_{RL} are the outputs of ICO learning and continuous actor–critic RL, respectively. ζ is a scale factor which is introduced to ensure that the sum is a valid control signal. The complete algorithm of combinatorial leaning with pseudocode is shown in Table 1.

In this learning scheme, ICO learning and continuous actor–critic RL can complement each other due to their learning principles. ICO learning [see Eq. (2)] relies on the predefined reflex signal, while continuous actor–critic RL [see Eq. (5)] depends on the TD error (δ) based on the estimated value function and the reward. As a consequence of the reflex avoidance learning principle, weight adaptation of ICO learning is initially more relevant than that from continuous actor–critic RL (until the value function is properly estimated). Thus, in some situations, like a pole balancing task (see Sec. 5.1), ICO learning quickly updates weights (i.e., control

Table 1. Combinatorial learning algorithm.

```
**Initialize** $\rho_k$, $w_k$, and $v_j$ to 0.0; $\varepsilon$ = Gaussian random number
**Repeat:**
At time step t
(1) observe reflex signal $x_0$ and sensory signals $x_k$ which are the state $\mathbf{x}$
(2) compute control output
        $O_{\text{ICO}} \leftarrow \rho_0 x_0 + \sum_{k=1}^{N} \rho_k x_k$
        $O_{\text{RL}} \leftarrow \varepsilon + \sum_{k=1}^{N} w_k x_k$
        $O_{\text{COM}} \leftarrow \zeta \cdot (O_{\text{ICO}} + O_{\text{RL}})$
(3) perform action
(4) observe reward $R$, new state $\mathbf{x}'$ and new reflex signal $x'_0$
(5) obtain value function by computing
        $a_j \leftarrow e^{-\|\mathbf{s}_j^T(\mathbf{x}-\mathbf{c}_j)\|^2}$
        $y_j \leftarrow \frac{a_j}{\sum_{l=1}^{M} a_l}$
        $V \leftarrow \sum_{j=1}^{M} v_j y_j$
(6) compute $\varepsilon \leftarrow \xi\sigma \cdot \min\left[1, \max\left[0, \frac{V_{\max}-V(\mathbf{x})}{V_{\max}-V_{\min}}\right]\right]$
(7) compute $\delta \leftarrow R + \gamma V(\mathbf{x}') - V(\mathbf{x})$
(8) update control parameters
        $\rho_k \leftarrow \rho_k + \mu x_k (x'_0 - x_0)$
        $w_k \leftarrow w_k + \alpha\delta x_k \varepsilon$
        $v_j \leftarrow v_j + \lambda\delta y_j$
**Until:** Successful control policy is found or the maximum number of trials
is reached.
```

parameters) to enhance or guide the entire learning process that includes continuous actor–critic RL. At the same time, ICO learning also utilizes the exploration strategy of continuous actor–critic RL to indirectly adapt its weights. In other situations, like a goal-directed behavior task (see Sec. 5.2), ICO learning plays roles on guiding continuous actor–critic RL to receive a reward and shaping a control policy. However, if reflex signal and TD error disagree, ICO learning may interfere with continuous actor–critic RL (see also Sec. 6 for more discussion on this).

Beside this, one important property of our approach is that we directly use sensory inputs as the state of a system (i.e., continuous state) without resorting to the explicit discretization of states and actions. Thus, this approach is capable of generating a continuous action leading to smooth control performance. It is also practical for continuous state-action problems (e.g., pole balancing and goal-directed behavior shown below) in particular in the domain of model-free control problems because of the learning rules (i.e., correlation-based learning and reward-based learning) which do not require a system or environment model.

5. Experiments and Results

We tested the performance of our combinatorial learning in two different tasks: A dynamic motion control task using a simulated cart-pole system and a goal-directed behavior control task using a simulated mobile robot system. In each of them we compared the performance of three control schemes: ICO learning, continuous actor–critic RL, and combinatorial learning. In addition, we also investigated interactions between ICO learning and continuous actor–critic RL by observing learning curves in order to understand their roles in combinatorial learning. It is important to note that the aim of this study is not to claim that the combination outperforms other/older methods for solving the tasks or model-free optimal control problems. Thus, comparing our combinatorial learning with other baseline methods (like, dynamic programming) will go beyond the scope of this work. Instead, we emphasize here that a combination is better than its individual components by utilizing the learning properties of ICO learning and continuous actor–critic RL.

5.1. *Dynamic motion control*

In this section, we demonstrate the performance of combinatorial learning (see Fig. 4) applied to a pole balancing problem [7] (see Fig. 5). The task was to balance an inverted pendulum, which is mounted on a cart moving freely in a one-dimensional interval, and to simultaneously avoid the interval boundaries. This cart-pole system was simulated on a desktop PC and updated by using a fourth-order Runge–Kutta method with a time step of 0.01 s. It provides four state variables: The angle of the pole with the vertical (θ), the pole angular velocity ($\dot{\theta}$), the position of the cart on the track (x), and the cart velocity ($\dot{x}$). Similar to Ref. [7], the cart was bound to move in the interval $-2.4 \leq x \leq 2.4\,[m]$ and the angle was

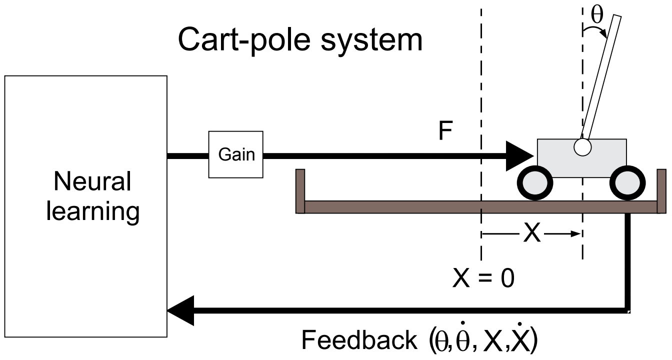


Fig. 5. Cart-pole system for a dynamic motion control task (see text for details).

allowed to vary in the interval $-12 \leq \theta \leq 12\,[°]$. The dynamics of the cart-pole system is modeled by:

$$\ddot{\theta} = \frac{g\sin\theta + \cos\theta\left(\frac{-F - ml\dot{\theta}^2\sin\theta + \mu_c \operatorname{sgn}(\dot{x})}{M+m}\right) - \frac{\mu_p\dot{\theta}}{ml}}{l\left(\frac{4}{3} - \frac{m\cos^2\theta}{M+m}\right)}, \tag{11}$$

$$\ddot{x} = \frac{F + ml(\dot{\theta}^2\sin\theta - \ddot{\theta}\cos\theta) - \mu_c \operatorname{sgn}(\dot{x})}{M+m}, \tag{12}$$

where $g = 9.8\,\text{m/s}^2$ denotes gravitational acceleration, $M = 1.0\,\text{kg}$ and $m = 0.1\,\text{kg}$ are mass of the cart and pole, respectively. $l = 0.5\,\text{m}$ is half of the pole length. $\mu_c = 5.0 \times 10^{-4}$ and $\mu_p = 2.0 \times 10^{-6}$ are friction coefficient of the cart and pole, respectively. F is a continuous force applied to the cart which is directly derived from the output of learning mechanisms with an amplifier gain of 10.0. Note that all these parameters and the cart-pole equations are generally used [7, 49].

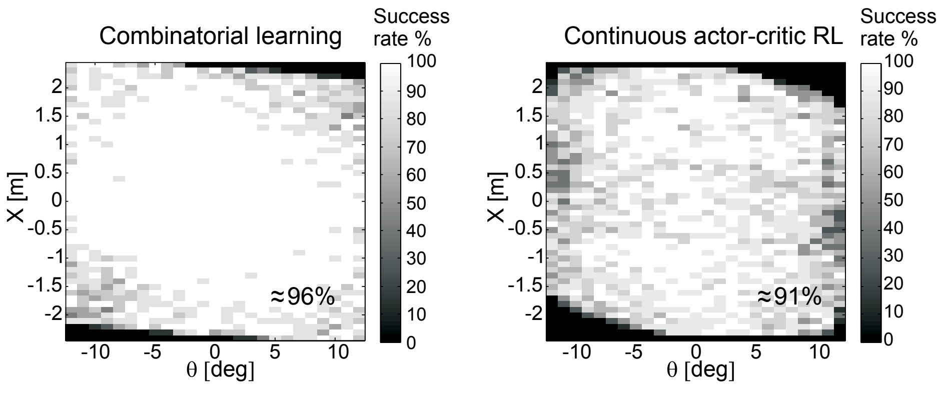
In fact, this task is difficult in its own right due to the limited boundaries of the pole angle and in particular the cart position. The boundaries are used as a standard benchmark setup in most control studies [7, 49]. In addition, its vertical upright equilibrium point to be balanced is inherently unstable (i.e., as any small disturbance may cause the pole to fall on the either side). From this setup, balancing the pole at critical initial conditions (e.g., $\theta = 11\,\text{deg}$, $x = 2.1\,\text{m}$) close to the boundaries is already difficult to find successful control policies by using a simple reward function.

In this setup, the four state variables (x, $\dot{x}$, θ, $\dot{\theta}$) of the system were used as sensory feedback ($x_{1,2,3,4}$) to ICO learning [see Eq. (1)] and continuous actor–critic RL [see Eqs. (3) and (7)]. For ICO learning, these state variables were scaled onto the interval $[-1, 1]$ similar to Ref. [49] and the reflex signal [x_0, see Eq. (1)] was given just before the system failed. The signal shows a positive activation (+1.0) if $x < -2.35\,\text{m}$ or $\theta > 11.5°$, a negative activation (−1.0) if $x > 2.35\,\text{m}$ or $\theta < -11.5°$, and 0 otherwise. Here, we set the learning rate [μ, see Eq. (2)] of ICO learning to 0.1. Note that the weights [$\rho_{1,2,3,4}$, see Eq. (2)] are changed only for the positive derivatives of the reflex signal, otherwise they remain unchanged. This is to avoid negative correlations resulting in poor performance.

For continuous actor–critic RL, we allocated three bases for x, three for $\dot{x}$, six for θ, and three for $\dot{\theta}$ according to a boxes approach (see Ref. [7] for more details). This leads to $3 \times 3 \times 6 \times 3 = 162$ bases employed as the centers of the critic network. Thus, the network has in total 162 hidden neurons [$M = 162$, see Eqs. (6) and (7)] which cover the state space of the system. The size or width of the Gaussian basis functions was simply set to twice the distance between its center and the center of its nearest neighbor. The reward signal [R, see Eq. (9)] was set to -1 at failure (i.e., cart hits the boundaries or pole falls to $\pm$ 12 deg) and 0 otherwise [7]. Here, $V_{\max}$ and $V_{\min}$ of the modulation scheme controlling the level of the exploration [ε, see Eq. (4)] were set to 0 and -1, respectively. In this setup, we set the scale factor (ξ) of the exploration to 5.0. This is to obtain a better performance. Thus, large changes of the weights of continuous actor–critic can occur. The control parameters α [see Eq. (5)], λ [see Eq. (8)], τ [see Eq. (9)], and ζ [see Eq. (10)] were set to 0.5, 0.5, 0.2, and 0.5, respectively.

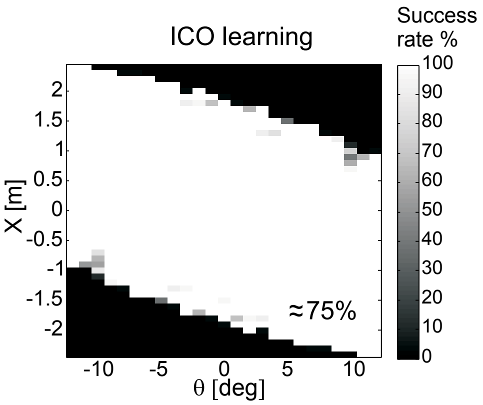
We let the combinatorial learning mechanism learn to balance the pole on 25×49 initial conditions (θ, x) while $\dot{\theta}$ and $\dot{x}$ were initially set to small random values using a Gaussian distribution with zero mean and a standard deviation of 0.1% of signal ranges which represents the system noise. Note that the control parameters (i.e., synaptic weights) of ICO learning and continuous actor–critic RL were initially set to 0.0. During a run each trial started with a given initial state and ended either in "success" (which occurs when the pole is kept in balance for at least 5×10^4 time steps or 500 s) or "failure" (which occurs when the pole falls 12 deg to either side or the cart moves 2.4 m to either side). Runs at each initial condition were terminated on failure or when a successful trial was achieved or the maximum number of trials was reached (here 1000 trials). The system was reset to the same initial state at failure. We repeated this for 25 experiments at each initial condition.

The performance of combinatorial learning is shown in Fig. 6(a). It can be seen that this learning mechanism was able to find successful control policies which can balance the pole and avoid the ends of the interval in a very large (x, θ)–domain of initial conditions [see Fig. 6(a)]. The system was successfully stabilized for $\approx 96\%$ of all initial conditions. This is because, on one hand, ICO learning utilizes the exploration strategy of continuous actor–critic RL to explore its parameter space such that a proper weight combination is obtained. At the same time continuous actor–critic RL is also guided by the built-in reflex of ICO learning to adapt its weights in a proper way. The remaining part (black areas), at which learning failed, is because of physical limitation. For example, if the system stands close to the right wall and the pole falls to the right, the cart momentum cannot be high enough to support the pole. Thus, it crashes into the wall before. The results we obtained here are comparable to the ones shown in Ref. [49] where this work employed



(a)

(b)



(c)

Fig. 6. Performance of three learning mechanisms for the pole balancing problem. (a) Successful control area (≈ 96%) of combinatorial learning on benchmark initial conditions. (b) Successful control area (≈ 91%) of continuous actor–critic RL. (c) Successful control area (≈ 75%) of ICO learning. Black areas represent a domain in which learning failed to solve the problem (i.e., it cannot learn to stabilize the system). A gray scale bar presents the success rate, i.e., the percentage of success from 25 experiments. Recall that "success" means the pole is kept in balance for at least 500 s.

similar linear control with four inputs but used an evolutionary algorithm for weight adaptation.

When only ICO learning or continuous actor–critic RL alone was applied, stabilizing the system was accomplished in smaller domains. For ICO learning alone, the system was balanced for ≈ 75% of all initial conditions [see Fig. 6(c)]. This is because ICO learning cannot explore the entire parameter space due to the lack of an exploration mechanism and it even cannot predict many steps into the future. It basically develops its weights (i.e., control parameters) with respect to an immediate correlation between predictive and reflex signals. In this setup, the built-in reflex occurs only at the last moment that the pole falls or the cart hits the wall.

Therefore, at initial conditions in which the system fails, the reflex signal cannot produce a strong cart momentum to turn the pole into an upright position or keep it balance for a certain period of time (i.e., avoiding the reflex). Thus, ICO learning cannot obtain a proper correlation between the predictive and reflex signals to achieve a proper weight combination. For continuous actor–critic RL alone, due to the lack of a prior knowledge, the system was balanced for $\approx 91\%$ of all initial conditions [see Fig. 6(b)]. This experimental result shows that, among the three learning mechanisms, combinatorial learning, which combines ICO learning and continuous actor–critic RL, was the best approach with respect to the success rate.

Note that, due mainly to the stochastic process of continuous actor–critic RL and partly to the introduced system noise which can easily destabilize the system, combinatorial learning and continuous actor–critic RL alone sometimes had difficulty or failed to find successful control policies in a given number of trials at, e.g., $x = -2.0\,\text{m}$, $\theta = 12\,\text{deg}$ and around $x = 0.0\,\text{m}$, $\theta = 0\,\text{deg}$, respectively. In contrast, ICO learning alone was almost 100% success at these initial conditions and even showed the very clear boundary between white and black areas [see Fig. 6(c)] since it is deterministic control where no exploration is involved.

To compare the learning speed of these three learning mechanisms in general cases, we observed their performance at a noncritical initial condition (e.g., $x = 1.0\,\text{m}$, $\theta = -1\,\text{deg}$), where they all can find successful control policies, and at a critical initial condition (e.g., $x = -1.8\,\text{m}$, $\theta = -5\,\text{deg}$) where combinatorial learning and continuous actor–critic RL can find the policies but ICO learning cannot. The result is shown in Fig. 7.

At the noncritical initial condition ICO learning was fastest, combinatorial learning was slower, and continuous actor–critic RL was the slowest [see Fig. 7(a)]. At the critical initial condition ICO learning failed while continuous actor–critic RL succeeded but required more learning trials compared to combinatorial learning. This experiment suggests that the fast convergence property of combinatorial learning is generally derived from ICO learning which can quickly learn to find a solution for a task but cannot properly learn solving a difficult task (i.e., here, stabilizing the system at a critical initial condition). Furthermore, the capability for solving a difficult task is basically obtained from continuous actor–critic RL which learns the task but usually takes many learning trials.

To better understand why the combined mechanism outperforms either ICO learning or continuous actor–critic RL alone, we also observed learning curves at a critical initial condition (e.g., $x = 2.1\,\text{m}$, $\theta = 11\,\text{deg}$) at which its individual components failed. Figure 8 shows that control parameters (i.e., synaptic weights) converged to fixed values when combinatorial learning was used [see thick lines in Figs. 8(a) and 8(b)]. This is because ICO learning and continuous actor–critic RL tried to find a proper weight combination such that a proper force is generated to push the cart for balancing the pole. This proper combination can be seen when the weight for one input in ICO learning increased and that in continuous actor–critic RL network decreased (e.g., ρ_x, w_x in Fig. 8).

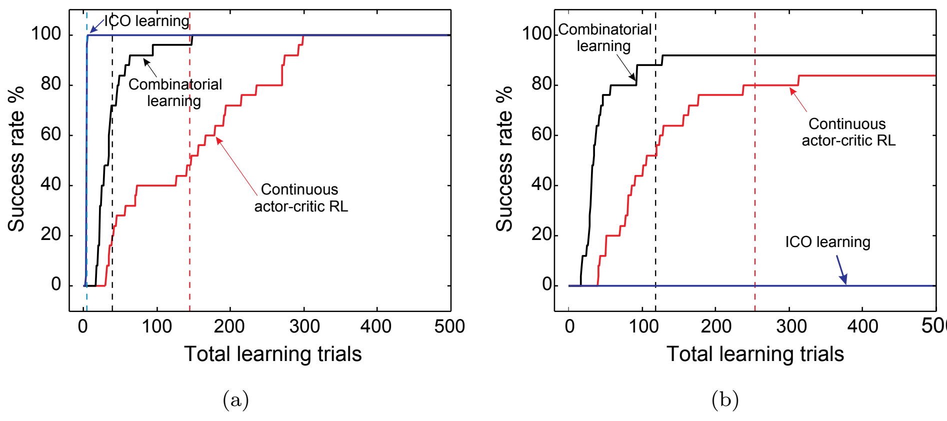
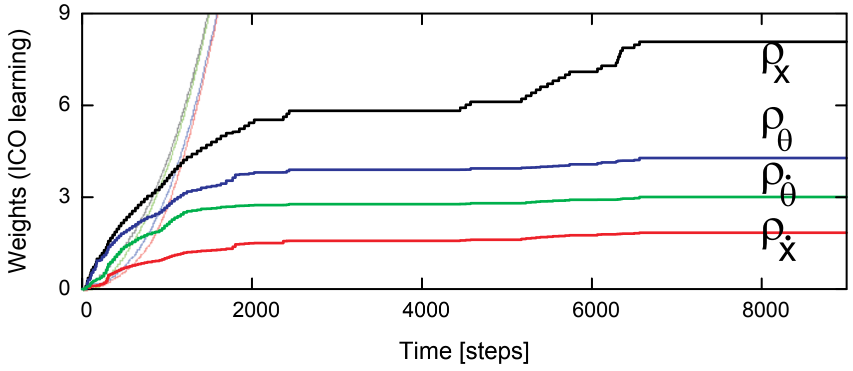


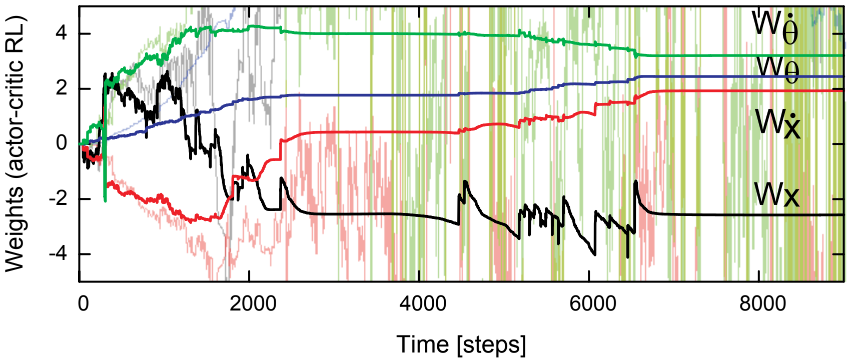
Fig. 7. (Color online) Comparison of the performance of three learning mechanisms. (a) Success rate according to total learning trials at the noncritical initial condition ($x = 1.0$ m, $\theta = -1$ deg). (b) Success rate according to total learning trials at the critical initial condition ($x = -1.8$ m, $\theta = -5$ deg). At this critical initial condition, ICO learning failed because during learning its weights grew more and more. Thus, the output of ICO learning, applied to the cart-pole system, also strongly increased. As a result, the output disturbed the system rather than balancing it. Note that we did not limit the output. Recall that success rate is calculated from the percentage of success in the total 25 experiments after a certain number of trials where "success" means the pole is kept in balance for at least 500 s. Dashed lines indicate the average of the total learning trials at success.

The behavior of the system controlled by all converged weights is shown in Fig. 9. Due to the proper weight combination, at the beginning a proper positive force was generated to push the cart to the right such that the pole could swing to the left to obtain an upward position. Afterwards, a negative force was generated to balance the pole and push the cart to the center. Finally, the system was stabilized at the center where all inputs were converged to zero values, thereby no force was generated. As a result, the pole was successfully balanced. If the converged weights of the ICO learning module were only used to control the system while the weights of the continuous actor–critic RL module were set to 0.0, the controller produced a very strong positive force to the cart at an early state. As a consequence, the pole fell to -12 deg (see Fig. 10). On the other hand, if the converged weights of the continuous actor–critic RL module were only used to control the system while the weights of the ICO learning module were set to 0.0, the controller produced a very strong negative force at an early state, thereby making the pole quickly fall to 12 deg (see Fig. 11). For these two cases, the system could not be stabilized since the forces were not properly generated.

When ICO learning alone [see transparent lines in Fig. 8(a)] was used to learn to balance the pole at the critical initial condition ($x = 2.1$ m, $\theta = 11$ deg), its control parameters diverged since the reflex signal cannot be avoided (i.e., the pole always fell). Although ICO learning is designed to learn to avoid a reflex signal, in this pole balancing setup the built-in reflex occurs only at the last moment that the pole falls



(a)



(b)

Fig. 8. (Color online) Learning curves at a critical initial condition ($x = 2.1\,\text{m}$, $\theta = 11\,\text{deg}$). (a) Weight changes in ICO learning. (b) Weight changes in continuous actor–critic RL. Thick lines present the weight changes in each learning mechanism in the combinatorial learning framework while transparent lines show the weight changes when only ICO learning or continuous actor–critic RL was used. Using combinatorial learning, the weights became stable after around 6500 time steps (or 70 trials) meaning that the system was successfully stabilized. In contrast, the weights diverged when only ICO learning [see (a)] was used while they changed a lot in the case of continuous actor–critic RL alone [see (b)]. Note that sudden change in w_x occurred (e.g., around 4400 steps) because there was a high correlation between the TD error and the input (x) while there were low correlations between the other inputs and the TD error. Here, $\rho_x = \rho_1$, $\rho_{\dot x} = \rho_2$, $\rho_\theta = \rho_3$, $\rho_{\dot\theta} = \rho_4$, $w_x = w_1$, $w_{\dot x} = w_2$, $w_\theta = w_3$, $w_{\dot\theta} = w_4$.

or the cart hits the wall. Therefore, in this difficult situation, a reflex signal cannot produce a strong cart momentum to turn the pole into an upright position or keep it balance for a certain period. Thus, ICO learning cannot obtain a proper correlation between the predictive and reflex signals; thereby its weights just increased more and more to try to find any proper weight combination. However, in this situation such a combination leading to a proper force cannot be achieved. While the weights were increasing, the output of ICO learning, applied to the cart-pole system, also

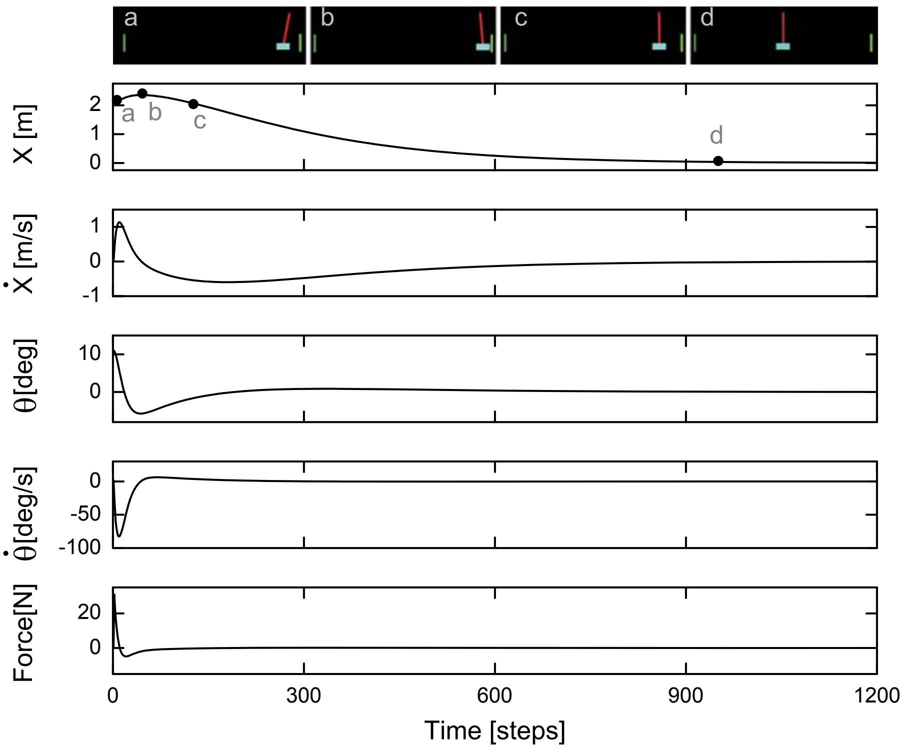


Fig. 9. States of the cart-pole system (x, $\dot{x}$, θ, $\dot{\theta}$) and the force under control of the learned weights ($\rho_x \approx 8.0774$, $\rho_{\dot{x}} \approx 1.8395$, $\rho_\theta \approx 4.2815$, $\rho_{\dot{\theta}} \approx 3.0069$, $w_x \approx -2.5790$, $w_{\dot{x}} \approx 1.9225$, $w_\theta \approx 2.4527$, $w_{\dot{\theta}} \approx 3.2216$, see e.g., Fig. 8) for the critical initial condition $x = 2.1$ m, $\theta = 11$ deg. A series of photos visualizing the cart-pole behavior at particular points is shown above.

increased. As a result, at some point the output disturbed the system rather than balancing it. In the case of continuous actor–critic RL alone [see transparent lines in Fig. 8(b)], the control parameters changed a lot due to the stochastic process employed which tried to search for a successful control policy. Note that at the early state of learning the weights of the ICO learning module in combinatorial learning became larger than the weights of ICO learning alone due to the stochastic process of the continuous actor–critic RL module in combinatorial learning. It can easily destabilize the system. Thus, the pole can often fall at the early state. This leads to the triggering of a reflex signal. On the other hand, in the case of ICO learning alone the pole fell less often at the early state such that the weights grew slower.

Finally, we investigated interactions between these two learning mechanisms. We first started one learning mechanism and then after a number of learning trials (e.g., 100 trials) we activated the other one (see Fig. 12). This is to observe two effects: (i) Can a later activated learning mechanism assist an earlier activated learning mechanism for policy improvement? and (ii) Can the earlier one provide

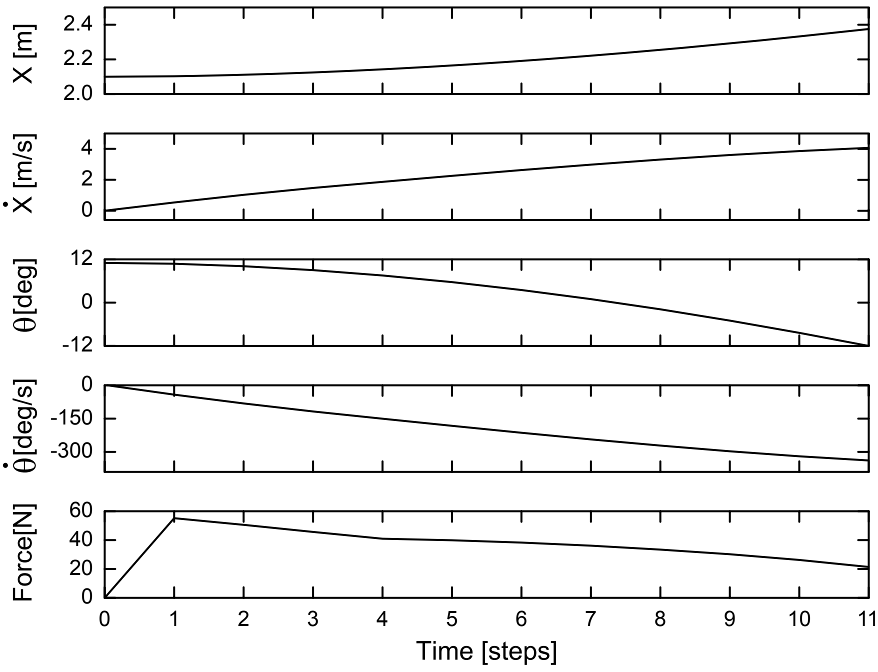


Fig. 10. States of the cart-pole system (x, $\dot{x}$, θ, $\dot{\theta}$) and the force under control of the learned weights ($\rho_x \approx 8.0774$, $\rho_{\dot{x}} \approx 1.8395$, $\rho_\theta \approx 4.2815$, $\rho_{\dot{\theta}} \approx 3.0069$, [see Fig. 8(a)] for the critical initial condition $x = 2.1$ m, $\theta = 11$ deg. We set w_x, $w_{\dot{x}}$, w_θ, and $w_{\dot{\theta}}$ to 0.0.

an appropriate developed control policy to the later one such that a successful control policy can still be achieved at the end?

Figures 12(a) and 12(b) show learning curves when continuous actor–critic RL was first started and followed by ICO learning after 100 trials [see dashed line in Fig. 12(a)]. It can be observed that after around 5500 time steps (or 130 trials), where ICO learning was already activated, the weight (w_x) of continuous actor–critic RL started to gradually change its growing direction into a different way [see e.g., thick line in Fig. 8(b)]. A similar effect also appears for the weight ($w_{\dot{x}}$) after around 9000 time steps (or 190 trials). This is because ICO learning can quickly find a correlation between a state and an unwanted condition (i.e., pole falls) and additionally generates the proper action when the pole falls through its built-in reflex. Thus, it can extract important features[c] serving to guide the learning strategy of continuous actor–critic RL. As a result, the weights of continuous actor–critic RL (e.g., w_x, $w_{\dot{x}}$) gradually changed to their appropriate directions but they did not change considerable compared to continuous actor–critic RL alone [see transparent lines in Fig. 8(b)].

[c]By feature we mean the combination between the weights and input signals.

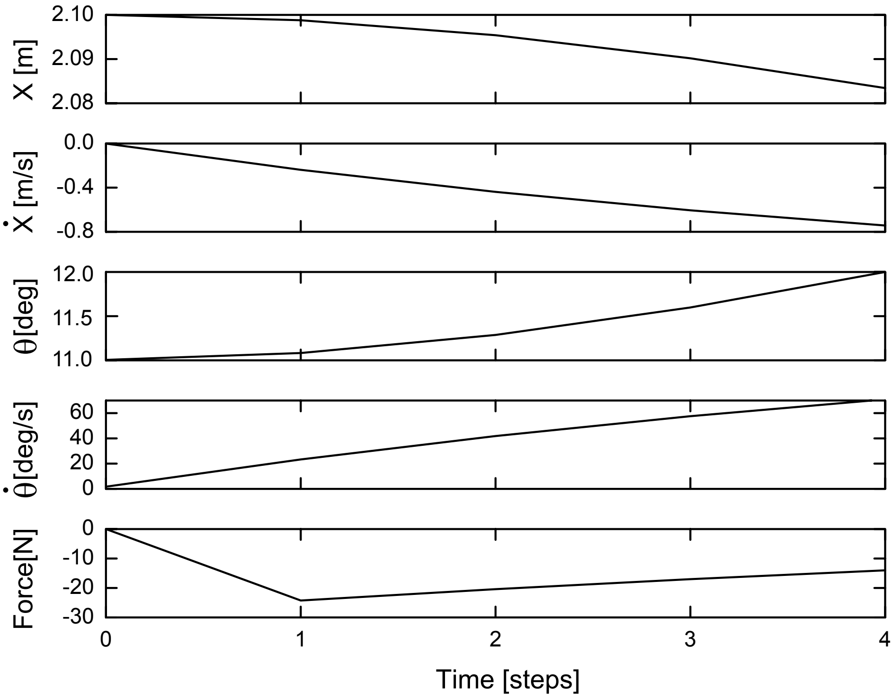


Fig. 11. States of the cart-pole system (x, $\dot{x}$, θ, $\dot{\theta}$) and the force under control of the learned weights ($w_x \approx -2.5790$, $w_{\dot{x}} \approx 1.9225$, $w_\theta \approx 2.4527$, $w_{\dot{\theta}} \approx 3.2216$, [see Fig. 8(b)]) for the critical initial condition $x = 2.1$ m, $\theta = 11$ deg. We set ρ_x, $\rho_{\dot{x}}$, ρ_θ, and $\rho_{\dot{\theta}}$ to 0.0.

Another interesting effect of the interaction is shown in Figs. 12(c) and 12(d) where ICO learning was first started and followed by continuous actor–critic RL after 100 trials [see dashed line in Fig. 12(d)]. After 115 trials (or around 2500 time steps), the pole did not fall anymore leading to reflex avoidance. As a consequence, the weights of ICO learning converged. However, the weights of continuous actor–critic RL still slightly changed due to the TD error. They finally converged (i.e., TD error ≈ 0) after around 17,600 time steps. This experiment shows that, on the one hand, continuous actor–critic RL seems to highly influence ICO learning such that the weights of ICO learning became stable shortly after continuous actor–critic RL was activated. On the other hand, ICO learning seems to provide an adequate control policy or an important feature to continuous actor–critic RL such that it can quickly adapt its weights to appropriate directions leading to convergence.

5.2. *Goal-directed behavior control*

Next, we present the performance of combinatorial learning (see Fig. 4) on a different task. Here, we employed it to a goal-directed behavior control problem. The task was to steer a wheeled mobile robot to move toward and finally approach a desired object (i.e., its goal) in a given time. In this scenario, we put the robot in

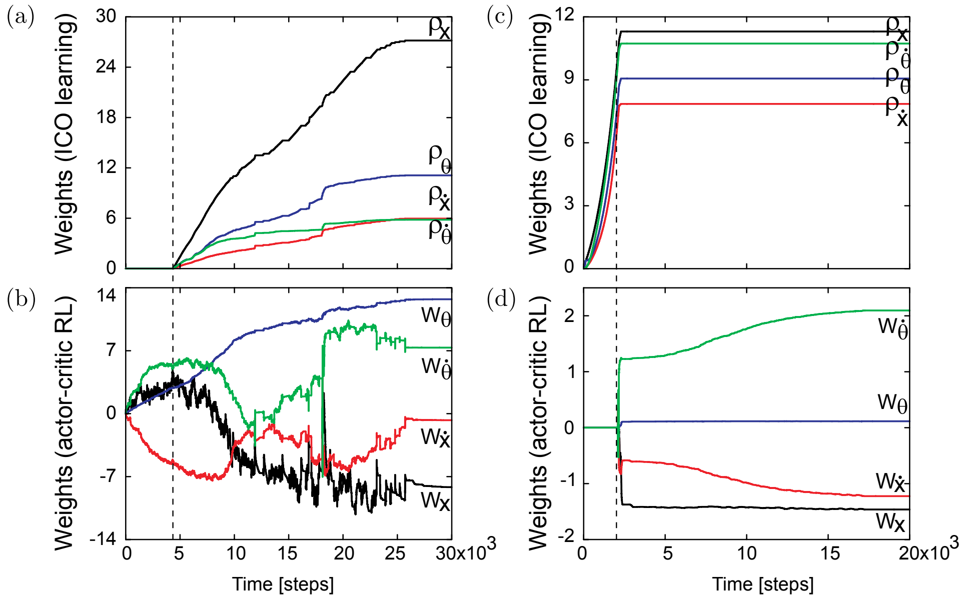


Fig. 12. (Color online) Learning curves at a critical initial condition ($x = 2.1$ m, $\theta = 11$ deg) when ICO learning and continuous actor–critic RL were not activated at the same time. (a), (b) Weight changes of ICO learning and continuous actor–critic RL in the combinatorial learning framework. In this experiment, ICO learning was activated after 100 trials (dashed line). (c), (d) Weight changes of ICO learning and continuous actor–critic RL where here continuous actor–critic RL was activated after 100 trials (dashed line).

a square area where one desired green object and one undesired blue object were provided. We used the physics simulator LPZROBOTS[d] to simulate the robot and its environment (see Fig. 13). The simulator was implemented on a desktop PC with an update time step of 0.01 s.

The mobile robot system provides four state variables, which are two relative orientations ($\phi_{G,B}$) and two relative positions ($D_{G,B}$) of the robot to the locations of the green (G) and blue (B) objects, and additional eight state variables of infrared (IR) sensors for boundary detection (see Fig. 13). $\phi_{G,B}$ provide information of how much the robot's direction deviates from the objects. They vary in the interval $[-180°, 180°]$ [see Fig. 13(b)] and show continuous values. If the objects are directly in front of the robot, $\phi_{G,B}$ show 0. If they are to the left-hand side of the robot, $\phi_{G,B}$ show negative values. If they are to the right-hand side of the robot, $\phi_{G,B}$ have positive values [see e.g., Fig. 13(b)]. $D_{G,B}$ provide information of how close the robot is to the objects. They are mapped onto the interval [0, 1], with 0 representing near, and $+1$ representing far. If the robot comes close to an object in a certain range [i.e., $D_{G,B} > 0.7$, see dashed areas in Fig. 13(c)], a reward is given for continuous

[d]It is based on the open dynamics engine (ODE) for more details of the LPZROBOTS simulator see http://robot.informatik.uni-leipzig.de/software/.

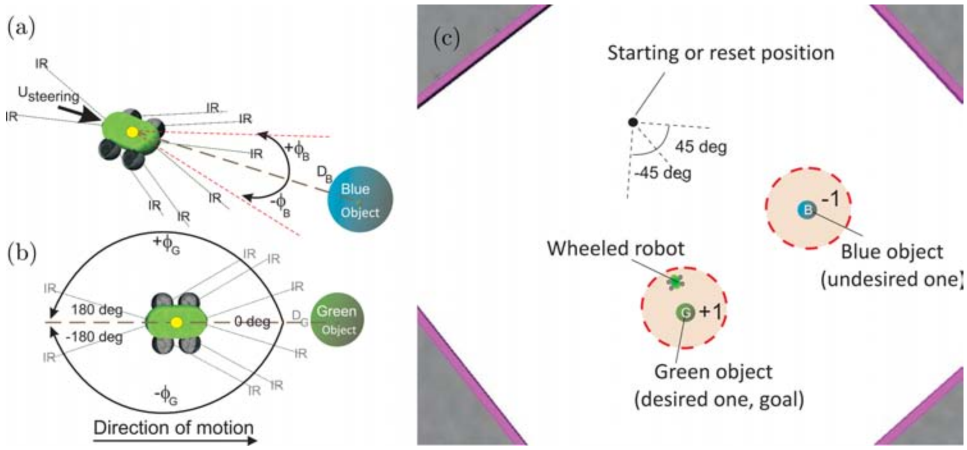


Fig. 13. (Color online) Simulated mobile robot system for a goal-directed behavior control task. (a) The mobile robot with different types of sensors (i.e., relative orientation ϕ and position D sensors and infrared IR sensors). (b) The variation of the relative orientation ϕ_G of the robot to the green object. (c) The environmental setup of the robot. The black dot represents the starting or reset position where the robot was initially started or reset after it hit a boundary or reached one of the objects. Dashed circles show areas where a positive (+1) or negative reward (−1) was given for continuous actor–critic RL and reflex signals were triggered for ICO learning (see text for details).

actor–critic RL and a reflex signal is triggered for ICO learning. The IR sensory signals are mapped onto the interval $[-1, +1]$, with -1 representing no boundary detection, and $+1$ representing hitting a boundary. This IR information was only used to reset the robot position on hitting a boundary.

It is important to note that in this setup, only $\phi_{G,B}$ were used as inputs (i.e., the state) to the control policy while $D_{G,B}$ were used only to generate reward and reflex signals for learning and to reset the robot position when approaching an object (i.e., $D_{G,B} > 0.95$). Thus, the robot has insufficient sensor data for reliably identifying its state in the environment. Furthermore, $\phi_{G,B}$ overlap with each other, i.e., the robot simultaneously senses its relative orientation to the locations of both objects in the whole area. Thus, both sensor signals try to steer the robot toward the corresponding objects once their synaptic weights have been developed.

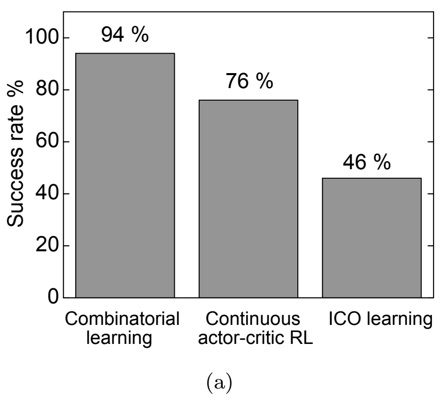
Here, for ICO learning, $\phi_{G,B}$ were used as predictive signals [$x_{1,2}$, see Eq. (1)]. Two independent reflex signals were configured: One was for the green object [x_{0_G}, see Eq. (1)] and the other for the blue one [x_{0_B}, see Eq. (1)]. They depend on the orientations ($\phi_{G,B}$) and the positions ($D_{G,B}$) of the robot to the objects. The reflex signals are triggered as soon as the robot comes close to the objects [i.e., entering areas inside the dashed circles as shown in Fig. 13(c)], and 0 otherwise. In fact, the reflex signals elicit a turn which is proportional to the deviations defined by $\phi_{G,B}$, i.e., the larger the deviations, the sharper the turn. Thereby, they turn the robot toward the objects. In other words, ICO learning tries to control the heading direction of the robot to align with an object. This way, it can implicitly optimize the behavior over the entire path. Since the green and blue objects are far from each other, the reflex areas do not overlap. Thus, the two reflex signals cannot be triggered at the same time. We set the learning rate [μ, see Eq. (2)] of ICO learning to 0.005. The weights [$\rho_{1,2}$, see Eq. (2)] were initially set to 0.0. They change only if the positive derivatives of the reflex signals are higher than a threshold, otherwise they remain unchanged. For example, when the robot comes close to the green object and the reflex signal is triggered, the weight (ρ_1) of the orientation signal with respect to the green object increases while the weight (ρ_2) of the blue one remains unaffected and vice versa when the robot comes close to the blue object.

For continuous actor–critic RL, we allocated four bases for ϕ_G and ϕ_B each. This leads to $4 \times 4 = 16$ bases employed as the centers of the critic network. Thus, the network has in total 16 hidden neurons [$M = 16$, see Eqs. (6) and (7)] which cover the state space of the system. The size or width of the Gaussian basis functions was simply set to twice the distance between its center and the center of its nearest neighbor. The reward signal [R, see Eq. (9)] was set to +1 when the robot came close to the green object (desired object or goal) and −1 to the blue object (undesired object). In order to promote exploration, we used low-pass filtered noise for low-frequency probing which was appropriate for the robot. We also used the modulation scheme for controlling the exploration level where ξ, $V_{\max}$ and $V_{\min}$ were here set to 5.0, 50 and 0, respectively. In addition to this scheme, the exploration term was exponentially reduced as soon as the performance improved (i.e., the robot frequently approached the goal). The control parameters α [see Eq. (5)], λ [see Eq. (8)], τ [see Eq. (9)], and ζ [see Eq. (10)] were set to 0.001, 0.7, 0.2, and 0.5, respectively. The weights [$w_{1,2}$, see Eq. (3)] were initially set to 0.0 and changed by Eq. (5).

We let the combinatorial learning mechanism learn to steer the robot to approach the desired goal (i.e., the green object). Without control, the robot randomly moved around. During a run in each trial, the robot started at a specific location [i.e., the black dot shown in Fig. 13(c)]. A run was terminated when the robot approached one of the objects or hit a boundary as well as when simulation time was above 15 s. After termination, the robot was reset to the same starting location with a random orientation in the interval [−45°, 45°]. We repeated this 50 experiments where each experiment was terminated after 200 trials. The performance of combinatorial learning compared to ICO learning and continuous actor–critic alone is shown in Fig. 14.

As can be seen, combinatorial learning had the highest success rate, continuous actor–critic RL a lower one, and ICO learning the lowest. With respect to the number of learning trials, combinatorial learning and ICO learning were not significantly different. However, they were substantially faster than continuous actor–critic RL. Among these learning mechanisms, combinatorial learning was the best approach, showing highest success rate with the lowest number of learning trials.

To better understand why the combinatorial learning mechanism outperforms its individual components in this task, we also plotted learning curves. Figure 15



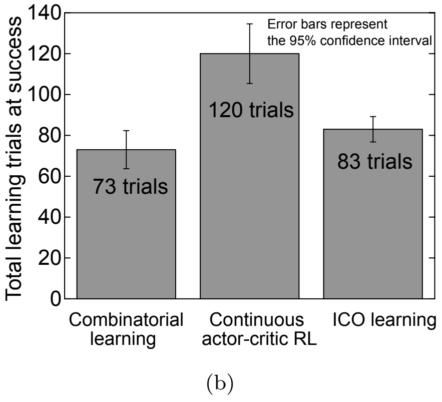
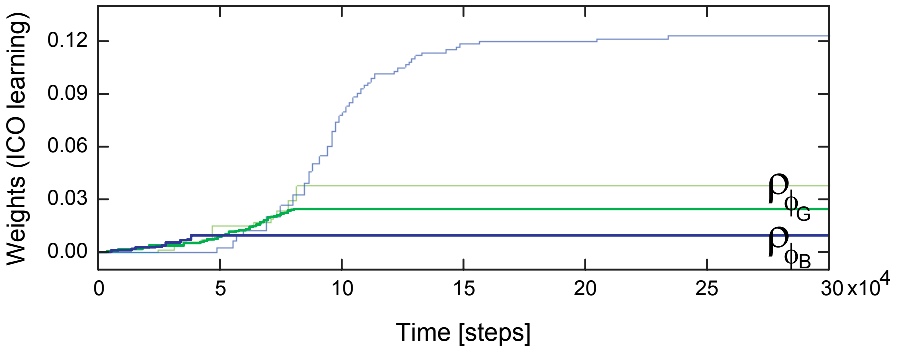


Fig. 14. Comparison of the performance of three learning mechanisms. (a) Success rate in a total of 50 experiments. Here, "success" means that the robot can approach the green object from the starting position with different random orientations in the interval $[-45°, 45°]$ [see e.g., Fig. 13(c)]. (b) Average of the total learning trials at success.

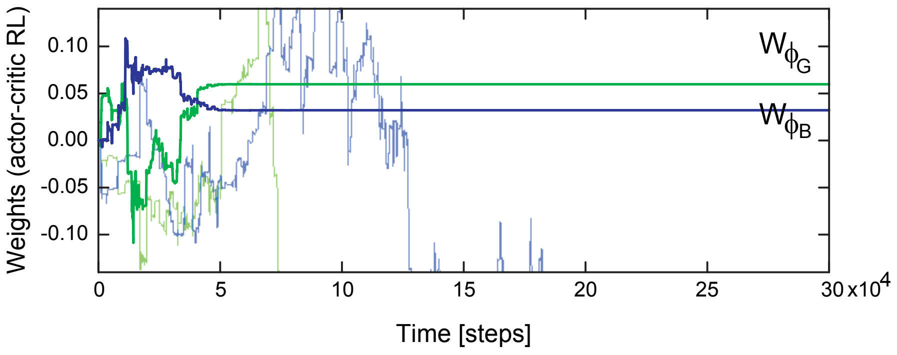
exemplifies the learning curves showing the changes of the control parameters (i.e., synaptic weights) of ICO learning and continuous actor–critic RL in combinatorial learning. The weights converged to fixed values [see thick lines in Figs. 15(a) and 15(b)] resulting in a goal-directed behavior. The input and output signals during this learning experiment and the behavior of the system after the weights converged are shown in Fig. 16.

When only ICO learning was used, the weights sometimes converged to other fixed values [see transparent lines in Fig. 15(a)] producing an undesired behavior; i.e., the robot moved toward the undesired blue object instead of the desired green object. When only continuous actor–critic RL was used, the weights sometimes changed a lot to negative values [see transparent lines in Fig. 15(b)]. As a consequence, the robot moved away from the objects. However, they will finally converge but this will require a lot of learning trials, e.g., > 600 trials.

In principle, ICO learning can recognize a correlation only between its inputs (i.e., predictive and reflex signals, see e.g., Fig. 1) without recognizing a goal (i.e., reward or punishment). Thus, for this task it can only generate an anticipatory reaction towards objects, rather than a goal-directed behavior. On the other hand, continuous actor–critic RL can achieve this in most cases but requires more learning trials than ICO learning. By contrast, combinatorial learning allows ICO learning and continuous actor–critic RL to complement each other leading to control policy improvement (high success rate and fast convergence [see Fig. 14]). This is because continuous actor–critic RL tries to drive a robot toward a goal with a certain degree of exploration. At the same time, ICO learning tries to limit the exploration area (i.e., guiding) since it tries to drive the robot toward the point of interest (green or blue object) defined by a prior knowledge. Without ICO learning, due to the exploration, the robot sometimes has difficulties to go back to the goal or it requires



(a)

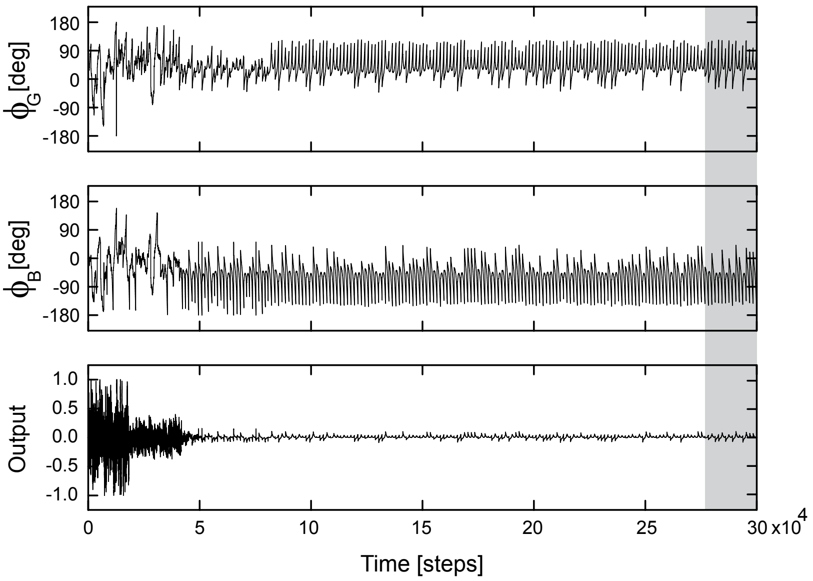


(b)

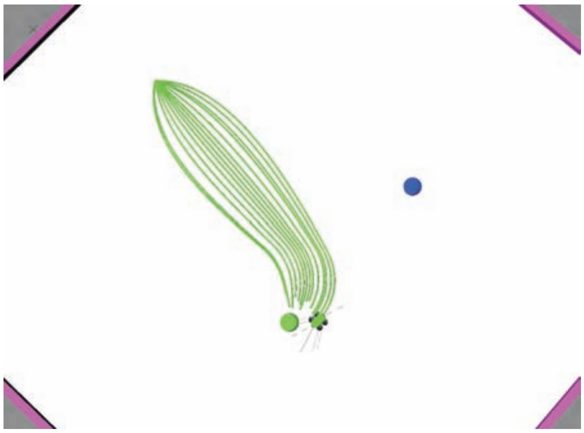
Fig. 15. (Color online) Learning curves of a goal-directed behavior. (a) Weight changes in ICO learning. (b) Weight changes in continuous actor–critic RL. Thick lines present the weight changes in each learning mechanism in the combinatorial learning framework while transparent lines show the weight changes when only ICO learning or continuous actor–critic RL was used. Using combinatorial learning, the weights became finally stable after around 75,000 time steps (or 50 trials) meaning that the robot can successfully approach the goal. In contrast, the weights sometimes converged to other fixed values in the case of ICO learning alone [see (a)] producing an undesired behavior (i.e., the robot went to a blue object) while they sometimes changed a lot in the case of continuous actor–critic RL alone [see (b)]. Note that here $\rho_{\phi_G} = \rho_1$, $\rho_{\phi_B} = \rho_2$, $w_{\phi_G} = w_1$, $w_{\phi_B} = w_2$.

more trials. In addition ICO learning can also shape the learning process such that the robot can approach the goal on a short path (see below).

To see the guiding and shaping effects from ICO learning, we took the developed weights of ICO learning and continuous actor–critic RL before convergence occurred to control the robot and observed its behavior. Then, we compared the behavior to the one controlled by only the developed weights of continuous actor–critic RL, i.e., we set the weights of ICO learning to 0.0 while the weights of continuous actor–critic RL remained unchanged. Interestingly, we found three different behaviors (I, II, III, see Figs. 17–19). Recall that in these goal-directed behavior experiments, the two relative orientations ($\phi_{G,B}$) overlap with each other, i.e.,



(a)



(b)

Fig. 16. (Color online) (a) States of the mobile robot system ($\phi_{G,B}$) and the output (O_{COM}) during learning. Learning curves belonging to these signals are shown in Fig. 15 (see thick lines). (b) Robot trajectories observed from around 28×10^4 to 30×10^4 time steps [see gray area in (a)]. Positive and negative values of the output means turning right and left, respectively. At the beginning the robot explored a lot (i.e., large amplitude of the output signal). After learning converged, the robot did not turn much (i.e., small amplitude of the output signal). It only turned if it deviated from the goal. As a result, it always approached the goal (green object). In other words, the learned control policy drove the robot toward the goal and kept it away from the blue object; thereby, ϕ_B shows most of the time negative values above 90 deg (i.e., heading away from the blue object) while ϕ_G shows most of the time positive values around 90 deg (i.e., turning towards the goal).

the robot simultaneously sensed its relative orientations to the locations of both objects in the whole area. In addition, for continuous actor–critic RL a positive reward (+1) was given when the robot got into the circle around the green object while a negative reward (−1) was given when the robot got into the circle around the blue object [see Fig. 13(c)]. Thus, during learning as long as the exploration term and the TD error existed, the weights of both signals simultaneously changed, no matter where the robot was.

Figure 17 shows the first behavior I where we took the developed weights at around 3×10^4 time steps [see dashed line in Figs. 17(a) and 17(b)], slightly before the weights of continuous actor–critic RL became stable, to test the robot. It can be seen that the robot always moved toward the desired green object [see Fig. 17(c)] when the developed weights of ICO learning and continuous actor–critic RL were used. On the other hand, it sometimes moved to the undesired blue object or went straight when only the developed weights of continuous actor–critic RL were used [see Fig. 17(d)] because of exploration as well as a large weight w_{ϕ_B}. Note that ρ_{ϕ_B}

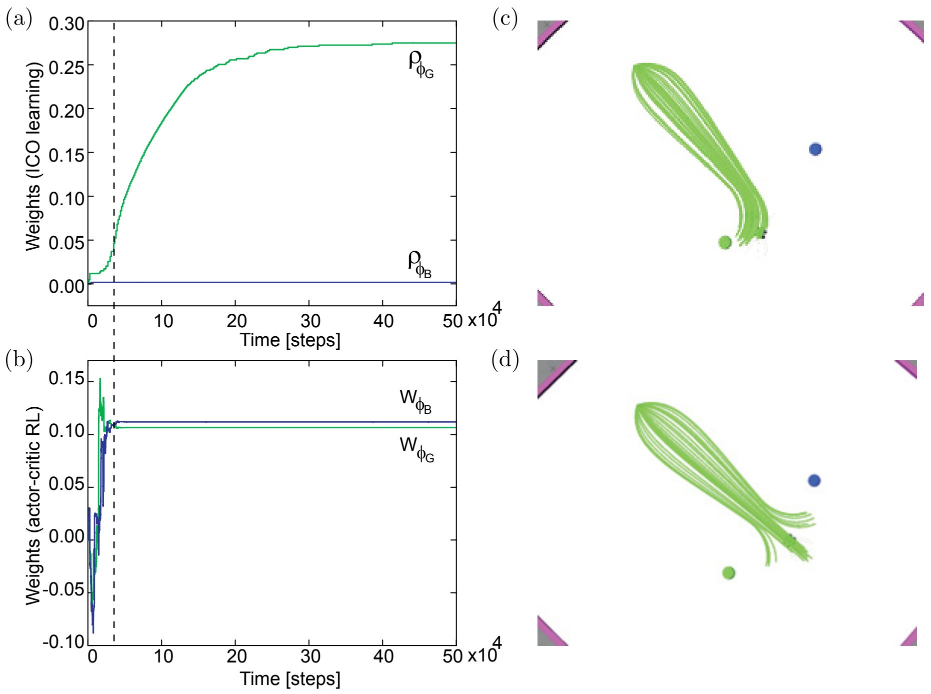


Fig. 17. (Color online) Learning curves and robot behaviors I. (a), (b) Weight changes of ICO learning and continuous actor–critic RL in the combinatorial learning framework. Dashed line shows the point (i.e., around 3×10^4 time steps or 300 s) where the weights of ICO leaning and continuous actor–critic RL were used to test the robot. (c) Trajectories of the robot from the starting position with different random orientations in the interval [−45°, 45°]. (d) The trajectories when ICO learning control policy was switched off, i.e., we set the weights of ICO learning to 0.0 while the weights of continuous actor–critic RL remained unchanged. Note that during the test we removed the exploration term from the controller in order to clearly see the trajectories.

did not become large since when the robot approached the blue object it did not deviate much from the object. Thus, the positive derivative of the reflex signal was smaller than threshold, thereby ρ_{ϕ_B} remained unchanged. This experimental result shows that ICO learning complemented continuous actor–critic RL leading to goal-directed behavior. In other words, ICO learning guided a learning strategy enabling continuous actor–critic RL to exploit more the positive reward. As a consequence, convergence finally occurred.

Figure 18 shows the second behavior II where we took the developed weights at around 20×10^4 time steps [see dashed line in Figs. 18(a) and 18(b)], slightly before the weights of continuous actor–critic RL reversed their growing directions, to test the robot. At this point, it can be seen that when the developed weights of ICO learning and continuous actor–critic RL were used the robot always approached the undesired blue object [see Fig. 18(c)] where the negative reward (-1) was given. Thus continuous actor–critic RL could use this reward signal to correct its current control policy. This effect can be observed from the weights of continuous

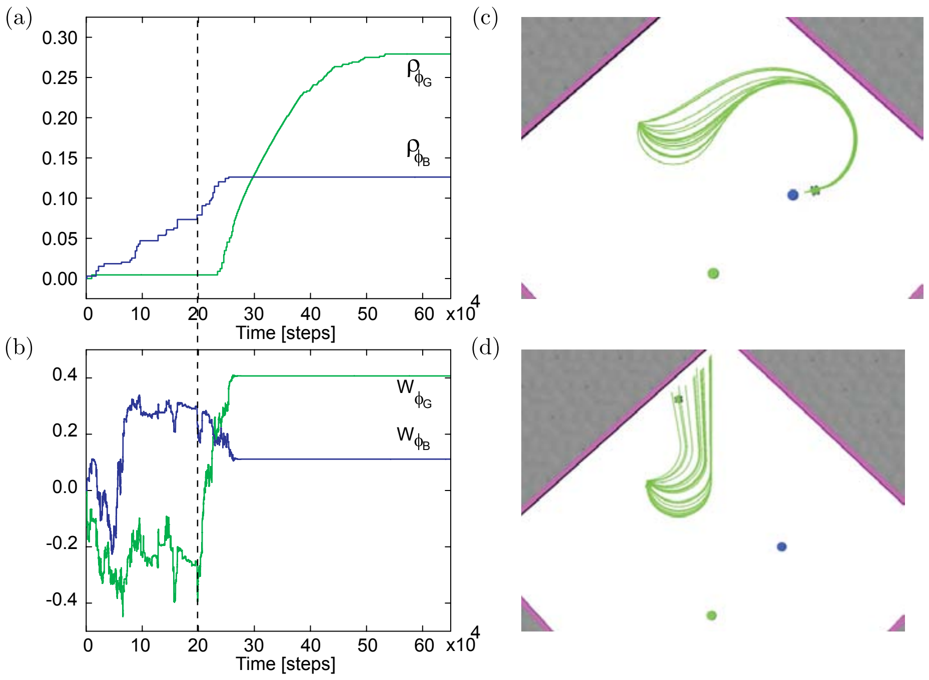


Fig. 18. (Color online) Learning curves and robot behaviors II. (a), (b) Weight changes of ICO learning and continuous actor–critic RL in the combinatorial learning framework. Dashed line shows the point (i.e., around 20×10^4 time steps or 2000 s) where the weights of ICO leaning and continuous actor–critic RL were used to test the robot. (c) Trajectories of the robot from the starting position with different random orientations in the interval $[-45^\circ, 45^\circ]$. (d) The trajectories when the ICO learning control policy was switched off, i.e., we set the weights of ICO learning to 0.0 while the weights of continuous actor–critic RL remained unchanged. Note that during the test we removed the exploration term from the controller in order to clearly see the trajectories.

actor–critic RL which significantly reversed their growing directions at around 20×10^4 time steps or 2000 s. On the other hand, when only the developed weights of continuous actor–critic RL were used, the robot always moved away from the objects [see Fig. 18(d)]. Therefore, in this situation continuous actor–critic RL had difficulty to obtain any reward signal to correct its current control policy. Due to the stochastic process employed, which tried to search for a successful control policy, the weights might change to a large degree [see transparent lines in Fig. 15(b)]. As a result, continuous actor–critic RL might fail to solve the task in a given number of trials (here, maximal 200 trials). This result suggests that ICO learning shaped or guided the learning strategy of continuous actor–critic RL such that it can receive a reward (i.e., here a negative one). Then it used this reward to correct the current control policy. As a consequence, convergence finally occurred.

Figure 19 shows the third behavior III where we took the developed weights at two states to test the robot. The early state was around 75×10^3 time steps where only the weights of continuous actor–critic RL became stable [see dashed line *State I* in Figs. 19(a) and 19(b)]. They were stable since the exploration term

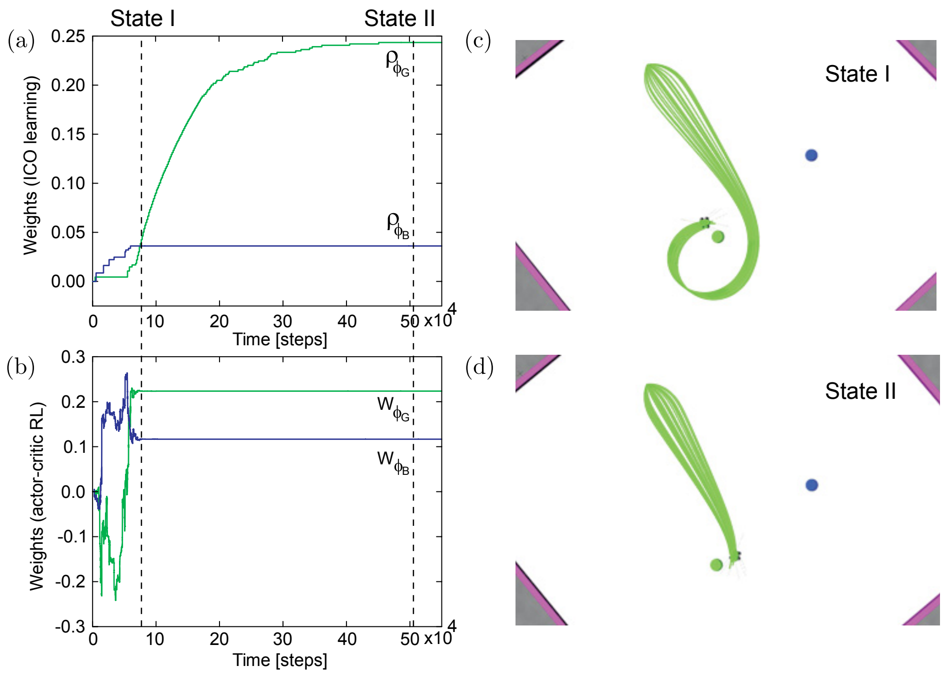


Fig. 19. (Color online) Learning curves and robot behaviors III. (a), (b) Weight changes of ICO learning and continuous actor–critic RL in the combinatorial learning framework. Dashed lines show two states where the weights of ICO leaning and continuous actor–critic RL were used to test the robot. State I was around 75×10^3 time steps or 750 s and state II was around 50×10^4 or 5000 s. (c) Trajectories of the robot from the starting position with different random orientations in the interval $[-45°, 45°]$ at state I. (d) The trajectories at state II. Note that during the test we removed the exploration term from the controller in order to clearly see the trajectories.

was zero. Recall that the exploration term was exponentially reduced as soon as the performance improved (i.e., the robot frequently approached the goal). The later state was around 50×10^4 time steps where the weights of ICO leaning became also stable [see dashed line *State II* in Figs. 19(a) and 19(b)]. It can be seen that the robot moved toward the goal in long trajectories [see Fig. 19(c)] when the control policy at the early state was used. In contrast, it moved on shorter trajectories when the control policy at the later state was used [see Fig. 19(d)]. This suggests that although continuous actor–critic RL was stopped due to the inhibition of its exploration term, ICO learning still shaped the control policy. This is because the reflex signal was not completely avoided since the robot still had large deviations to the goal when it came close to it. As a consequence, ICO learning improved robot performance by making it head directly to the goal, thereby leading to shorter trajectories.

It is important to note that although the resulting weights of the experiments shown in Figs. 15 and 17–19 converged to different values, they generally converged to almost the same weight ratio ($\frac{\rho_{\phi_G} + w_{\phi_G}}{\rho_{\phi_B} + w_{\phi_B}}$) of 2.9 ± 0.6. This shows that in combinatorial learning the combination of the weights of these two modules is necessary to successfully solve the task. Using all learned weights even yields a better result (i.e., the robot moved on the shortest trajectories) compared to using only the learned weights of either the ICO learning module or the continuous actor–critic RL module (see Fig. 20). In addition, the weight ratio also suggests that the positive reward attracts the system approximately three times larger than the negative reward repulses it.

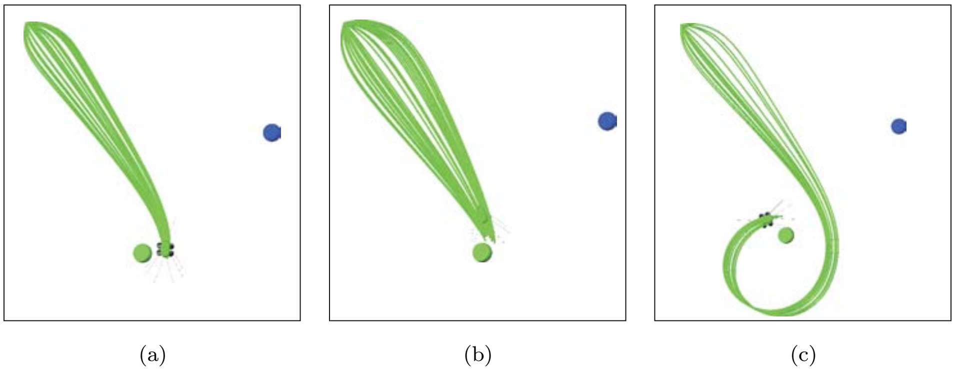


Fig. 20. (Color online) Robot behaviors at three different control parameter setups of combinatorial learning. For our investigation here, we used the learned weights from the experiment shown in Fig. 19, i.e., the weights at State *II*. (a) All learned weights were used, i.e., $\rho_{\phi_G} \approx 0.245$, $\rho_{\phi_B} \approx 0.036$, $w_{\phi_G} \approx 0.224$, $w_{\phi_B} \approx 0.117$. (b) Only learned weights of the ICO learning module were used while the weights of the continuous actor–critic RL modules were set to 0.0. (c) Only learned weights of the continuous actor–critic RL module were used while the weights of the ICO learning module were set to 0.0. Note that in each test the robot started from the starting position with different random orientations in the interval $[-45°, 45°]$ and we removed the exploration term from the controller in order to clearly see the trajectories.

6. Conclusion

In the following, we will discuss some remaining issues while other relevant discussion points have been treated alongside the experimental section above.

In this study, we introduced a neural combinatorial learning model for policy improvement. The learning model combines ICO learning and continuous actor–critic RL in a parallel manner where the ICO learning output and the continuous actor–critic RL output are equally weighted to control the agent. The equal contribution used here is a simple and straightforward strategy for combining them [see Eq. (10)]. In general, ICO learning alone can quickly learn to solve tasks but has limitations for more difficult tasks (i.e., here, balancing the pole in critical initial conditions as well as goal-directed behavior). On the other hand, pure continuous actor–critic RL can often solve the tasks but learns slowly.

Mainly we found that the performance of the controller can be strongly improved when the combinatorial learning model was applied. To make this model work properly we need to design a proper reflex for ICO learning as well as an appropriate correlation between predictive and reflex signals. For example, in the pole balancing task we configured ICO learning such that the weights were changed only for the positive derivatives of the reflex signal, otherwise they remained unchanged. This is to avoid a negative correlation resulting in poor learning performance or even failure. Another condition which would make the current form of the model problematic is a strong conflict between the reward function and the reflex. Some problematic cases would be:

(i) The first case is if continuous actor–critic RL moves the agent toward a target due to the reward function, while ICO learning tries to move it away from it due to the built-in reflex.
(ii) Another case is robot navigation in an environment with obstacles when ICO learning is used to generate a negative tropism behavior (e.g., avoiding obstacles) while continuous actor–critic RL is used to generate a positive tropism behavior (e.g., approaching a goal). In this scenario, a conflict will occur when the goal is behind an obstacle or directly close to it.
(iii) The last case is a dynamic motion control task like balancing a humanoid robot (many degrees of freedom system) against an external disturbance (pushing) where ICO learning controls the robot to avoid pushing (e.g., leaning action) while continuous actor–critic RL wants to keep it balance (e.g., upright position).

However, these complex tasks might also be solved but this will require a modification of the model or an improvement by:

(i) Properly designing the reward function of continuous actor–critic RL and the reflex of ICO learning,
(ii) Using more appropriate sensory signals or predictive signals,

(iii) Transforming the low-dimensional input of the actor part into a higher-dimensional one by using nonlinear functions (e.g., an RBF network [46]) or using a decoder [7],
(iv) Using an adaptive critic network or another type of a critic network (e.g., a self-adaptive reservoir computing network [14] having high-dimensional non-linear dynamics and internal memory) for a better approximation of the value function. These issues (i–iv) are still under investigation and go beyond the scope of this paper.

Besides our approach, there are a number of investigations on combining conventional RL with other (learning) mechanisms or applying other adaptive methods to it in order to enhance learning capability, reduce learning time, or counteract the curse of dimensionality. For example, Price and Boutilier [54] developed an imitation model called "implicit imitation" and integrated it into RL. It basically combines its own experience with its observations of the behavior of an expert mentor for learning. Doya [18] proposed hybrid RL based on the "actor-tutor" framework, which uses a model of the system dynamics as the tutor part. There, the actor (or controller) is trained by supervised learning to minimize the difference between its output and the tutor's output (desired output). This framework, basically resembling "feedback error learning" [23], was applied to nonlinear control tasks. Centina [12] introduced a supervised reinforcement learning (SRL) architecture for robot control problems with high-dimensional state spaces. There, a behavior model learned from examples is used to dynamically reduce the set of actions available from each state during the early RL process. In addition to these, other efforts have been made by developing advanced RL techniques [21, 31, 58] like hierarchical RL [3, 6, 9, 17, 66], by employing adaptive state representation [28, 34, 57], by using different exploration/exploitation techniques [4, 44, 60, 64], and by introducing algorithms for shaping rewards [2, 16, 48]. While all these advanced methods can successfully solve several (robot) tasks and are effective in their own right, they are quite difficult to match to biological neural learning and conditioning paradigms.

Only a few works have developed different types of learning models for robot control where the models mimic principles of these learning or conditioning mechanisms of animal leaning [1, 13, 65]. Alonso *et al.* [1] introduced the associative learning based approach (called the Pavlovian and Instrumental Q-learning framework) to deal with generalization in Q-learning. This approach improves RL (i.e., Q-learning) by applying the Rescorla-Wagner model [56] as a part of the control scheme for stimulus-stimulus associations. Its performance was tested in a grid simulator where agents have to approach or avoid appetitive and aversive stimuli. Chang and Gaudiano [13] presented a neural network based on operant and classical conditioning. It was tested on mobile robots. As a consequence, it allows the robots to simultaneously learn to approach light sources and avoid obstacles. Touretzky and Saksida [65] developed a model of operant conditioning that incorporates aspects of chaining in which behavioral routines are built up from smaller action segments.

The model was implemented on a mobile robot for solving the delay match to sample task (i.e., the task that involves behavioral sequences). Although all these learning models [1, 13, 65] employed learning and conditioning aspects of animal learning, they did not show or provided an understanding of how different mechanisms interact or complement each other, resulting in successful control policies. Instead, they were developed for improving conventional RL models or solving particular robotic tasks (i.e., goal-directed behavior control). Therefore, it is still unclear whether these models can also deal with qualitatively different tasks, like dynamic motion control.

Compared to many of these approaches just summarized, our combinatorial learning model applied the principles of classical and operant conditioning of animal learning. It was developed using ICO learning (a simplified model of classical conditioning) and continuous actor–critic RL (a simplified model of operant conditioning). They were implemented based on artificial neural networks; thereby, making them conceptually closer to biological systems compared to any other solution. Furthermore, ICO learning and continuous actor–critic RL are partially related to neural learning mechanisms in the brain. Specifically, ICO learning implements plain heterosynaptic plasticity associated with modulatory processes found in the brain [27] and continuous actor–critic RL uses TD learning which is related to dopaminergic responses in the brain. Especially, some cells in the substantia nigra and ventral tegmental area (VTA) show a behavior similar to representing the error of TD-learning [61], which we use for weight adaptation in actor and critic networks. We demonstrated the capability of this model in solving two different tasks: Pole balancing and goal-directed behavior control. This shows that the learning model is not limited to a specific task.

In addition to this, our model can be considered as a model-free method since its learning rules do not require a system or environment model. Instead, ICO learning requires only a built-in reflex as a self-supervised mechanism to quickly find the correlations between a state and an unwanted condition (i.e., reflex action), while continuous actor–critic RL uses its prediction mechanism including its own experiences and some exploration to obtain a good control policy. Although our models use a fixed state representation in the critic, one could also extend the critic to adaptive state partitioning [45] since the actor and critic are independently constructed. Our work also shares a connection to Kolter and Ng [33] where they presented a policy gradient method called the "Signed Derivative" approximation. The general concept of this approach is similar to our model in the sense that it is a model-free method which uses intuition to guess the direction where control inputs affect future state variables. This intuition is used to construct the signed derivative approximation which is directly applied to the update rule of RL. Generally speaking, the signed derivative approximation can be viewed as an instance of the built-in reflex of ICO learning. However, in our model the built-in reflex indirectly affects RL through ICO learning since it is used to guide and shape learning.

In summary, the study pursued here sharpens our understanding of how different learning mechanisms (i.e., correlation-based learning and RL) can be appropriately combined and how they complement each other leading to policy improvement. This study also suggests that correlation-based learning, on the one hand, can be used to speed up the learning process of RL. On the other hand, it can shape and correctly guide RL for searching an optimal policy. While the proposed combination of these two learning mechanisms can improve the performance of the systems, they are still combined in a simple way [see Eq. (10)]. Thus, for future work, we will investigate adaptive combinations. One possible option is to employ a learning mechanism based on a correlation between a direct reward signal and the outputs of ICO learning and continuous actor–critic RL for adapting their output weights. This way, an active output will have a high correlation with the reward signal, thereby strengthening its weight. The output weights will finally determine the behavior of the agent. Another option is to use a hierarchical RL framework [46] to find an optimal combination. Furthermore, we will also apply the combinatorial learning mechanism to more complex tasks, like the double-pendulum scenario [22, 29, 30], including ones with high dimensional states and actions (e.g., helicopter control [36] and octopus arm problems [69]). We also aim to use it as online learning for real robotic tasks, e.g., adaptive walking of hexapod robots [41], dynamic motion control of biped robots, and real robot navigation in complex environments. However, solving such tasks may require a modification of some components, e.g., using a nonlinear actor and/or an adaptive critic network, which could be easily done due to the modularity of the framework.

Acknowledgments

This research was supported by Emmy Noether grant MA4464/3-1 of the Deutsche Forschungsgemeinschaft (DFG), Bernstein Center for Computational Neuroscience II Göttingen (BCCN grant 01GQ1005A, project D1), Japan Society for the Promotion of Science (JSPS), European Communitys Seventh Framework Programme FP7/2007-2013 (Theme 3, Information and Communication Technologies) under grant agreement 270273, Xperience, and a part of this research was supported by "Brain Machine Interface Development" SRPBS, MEXT, MEXT KAKENHI 23120004 and Strategic International Cooperative Program, JST. P.M. would like to thank NICT for its support within the JAPAN TRUST International Research Cooperation Program. We thank Tomas Kulvicius for critical discussions and Martin Biehl and Frank Hesse for technical advice.

References

[1] Alonso, E., Mondragon, E. and Kjäll-Ohlsson, N., Pavlovian and instrumental Q-learning: A Rescorla and Wagner-based approach to generalization in Q-learning, in *Proc. Adaptation in Artificial and Biological Systems* (2006), pp. 23–29.

[2] Asmuth, J., Littman, M. L. and Zinkov, R., Potential-based shaping in model-based reinforcement learning, in *Proc. 23rd National Conf. Artificial Intelligence* (*AAAI'08*) (2008), pp. 604–609.

[3] Bakker, B. and Schmidhuber, J., Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization, in *Proc. 8th Conf. Intelligent Autonomous Systems* (2004), pp. 438–445.

[4] Banerjee, B. and Kraemer, L., Action discovery for single and multi-agent reinforcement learning, *Adv. Complex Syst.* **14** (2011) 279–305.

[5] Barnard, C., *Animal Behavior*: *Mechanism*, *Development*, *Function*, *and Evolution* (Pearson Education, 2004).

[6] Barto, A. G. and Mahadevan, S., Recent advances in hierarchical reinforcement learning, *Discrete Event Dyn. Syst.* **13** (2003) 41–77.

[7] Barto, A. G., Sutton, R. S. and Anderson, C. W., Neuron-like adaptive elements that can solve difficult learning control problems, *IEEE Trans. Syst. Man*, *Cybern.* **13** (1983) 834–846.

[8] Bekey, G., *Autonomous Robots From Biological Inspiration to Implementation and Control* (MIT Press, Cambridge, 2005).

[9] Botvinick, M. M., Niv, Y. and Barto, A. C., Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective, *Cognition* **113** (2009) 262–280.

[10] Bovet, S., Robots with self-developing brains, Ph.D. thesis, University of Zurich (2007).

[11] Brembs, B. and Heisenberg, M., The operant and the classical in conditioned orientation in drosophila melanogaster at the flight simulator, *Learn. Memory* **7** (2000) 104–115.

[12] Cetina, V. U., Supervised reinforcement learning using behavior models, in *Proc. Sixth Int. Conf. Machine Learning and Applications* (*ICMLA 2007*) (2007), pp. 336–341.

[13] Chang, C. and Gaudiano, P., Application of biological learning theories to mobile robot avoidance and approach behaviors, *Adv. Complex Syst.* **1** (1998) 79–114.

[14] Dasgupta, S., Wörgötter, F. and Manoonpong, P., Information theoretic self-organised adaptation in reservoirs for temporal memory tasks, in *Proc. 13th Int. Conf. Engineering Applications of Neural Networks* (*EANN 2012*) (2012), pp. 31–40.

[15] Dayan, P. and Balleine, B., Reward, motivation, and reinforcement learning, *Neuron* **36** (2002) 285–298.

[16] Devlin, S., Kudenko, D. and Grzes, M., An empirical study of potential-based reward shaping and advice in complex, multi-agent systems, *Adv. Complex Syst.* **14** (2011) 251–278.

[17] Dietterich, T. G., Hierarchical reinforcement learning with the MAXQ value function decomposition, *J. Artif. Intell. Res.* **13** (2000) 227–303.

[18] Doya, K., Efficient nonlinear control with actor-tutor architecture, in *Advances in Neural Information Processing Systems* (1997), pp. 1012–1018.

[19] Doya, K., Reinforcement learning in continuous time and space, *Neural Comput.* **12** (2000) 219–245.

[20] Endo, G., Morimoto, J., Matsubara, T., Nakanish, J. and Cheng, G., Learning CPG-based biped locomotion with a policy gradient method: Application to a humanoid robot, *Int. J. Robot. Res.* **27** (2008) 213–228.

[21] Fischer, J., *A Modulatory Learning Rule for Neural Learning and Metalearning in Real World Robots with Many Degees of Freedom* (Shaker Verlag GmbH, 2003).

[22] Gomez, F., Schmidhuber, J. and Miikkulainen., R., Accelerated neural evolution through cooperatively coevolved synapses, *J. Mach. Lear. Res.* **9** (2008) 937–965.

[23] Gomi, H. and Kawato, M., Neural network control for a closed-loop system using feedback-error-learning, *Neural Netw.* **6** (1993) 933–946.

[24] Gullapalli, V., A stochastic reinforcement learning algorithm for learning real-valued functions, *Neural Netw.* **3** (1990) 671–692.

[25] Howery, L. D., Why do animals behave the way they do?, *Backyards and Beyond*: *Rural Living in Arizona* **3** (2007) 17–18.

[26] Hull, C. L., *A Behavior System*: *An Introduction to Behavior Theory Concerning the Individual Organism* (Yale University Press, New Haven, CT, 1952).

[27] Humeau, Y., Shaban, H., Bissiere, S. and Lüthi, A., Presynaptic induction of heterosynaptic associative plasticity in the mammalian brain, *Nature* **426** (2003) 841–845.

[28] Iida, S., Kuwayama, K., Kanoh, M., Kato, S. and Itoh, H., A dynamic allocation method of basis functions in reinforcement learning, in *Proc. 17th Australian Joint Conf. Artificial Intelligence* (2004), pp. 71–73.

[29] Kassahun, Y., de Gea, J., Edgington, M., Metzen, J. H. and Kirchner, F., Accelerating neuroevolutionary methods using a kalman filter, in *Proc. 10th Genetic and Evolutionary Computation Conf.* (*GECCO-2008*) (2008), pp. 1397–1404.

[30] Kassahun, Y., Wöhrle, H., Fabisch, A. and Tabie, M., Learning parameters of linear models in compressed parameter space, in *Proc. Artificial Neural Networks and Machine Learning* (*ICANN2012*) (2012), pp. 108–115.

[31] Kawarai, N. and Kobayashi, Y., Learning of whole arm manipulation with constraint of contact mode maintaining, *J. Robot. Mechatron.* **22** (2010) 542–550.

[32] Klopf, A. H., A neuronal model of classical conditioning, *Psychobiology* **16** (1988) 85–123.

[33] Kolter, J. and Ng, A., Policy search via the signed derivative, in *Proc. Robotics*: *Science and Systems* (*RSS*) (2009), pp. 27, Online.

[34] Kondo, T. and Ito, K., A reinforcement learning with adaptive state space recruitment strategy for real autonomous mobile robots, in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (2002), pp. 897–902.

[35] Konorski, J., *Integrative Activity of the Brain* (University of Chicago Press, Chicago, 1967).

[36] Koppejan, R. and Whiteson, S., Neuroevolutionary reinforcement learning for generalized helicopter control, in *GECCO 2009*: *Proc. Genetic and Evolutionary Computation Conf.* (2009), pp. 145–152.

[37] Kosco, B., Differential hebbian learning, in *Proc. Neural Networks for Computing*: *AIP*, Vol. 151 (1986), pp. 277–282.

[38] Lee, H., Shen, Y., Yu, C., Singh, G. and Ng, A., Quadruped robot obstacle negotiation via reinforcement learning, in *Proc. IEEE Int. Conf. Robotics and Automation* (2006), pp. 3003–3010.

[39] Lovibond, P. F., Facilitation of instrumental behavior by a pavlovian appetitive conditioned stimulus, *J. Exp. Psychol. Anim. B* **9** (1983) 225–247.

[40] Manoonpong, P., Geng, T., Kulvicius, T., Porr, B. and Wörgötter, F., Adaptive, fast walking in a biped robot under neuronal control and learning, *PLoS Comput. Biol.* **3** (2007) e134.

[41] Manoonpong, P., Parlitz, U. and Wörgötter, F., Neural control and adaptive neural forward models for insect-like, energy-efficient, and adaptable locomotion of walking machines, *Front. Neural Circuits* **7** (2013). Doi: 10.3389/fncir.2013.00012.

[42] Manoonpong, P. and Wörgötter, F., Adaptive sensor-driven neural control for learning in walking machines, in *Neural Information Processing*, *LNCS* (2009), pp. 47–55.

[43] Manoonpong, P., Wörgötter, F. and Morimoto, J., Extraction of reward-related feature space using correlation-based and reward-based learning methods, in *Neural Information Processing*, *LNCS* (2010), pp. 414–421.

[44] Morihiro, K., Matsui, N., and Nishimura, H., Effects of chaotic exploration on reinforcement maze learning, in *Knowledge Based Intelligent Information and Engineering Systems* (2004), pp. 833–839.

[45] Morimoto, J. and Doya, K., Reinforcement learning of dynamic motor sequence: Learning to stand up, in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (1998), pp. 1721–1726.

[46] Morimoto, J. and Doya, K., Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning, *Robot. Auton. Syst.* **36** (2001) 37–51.

[47] Mowrer, O., *Learning Theory and Behavior* (New York, Wiley, 1960).

[48] Ng, A. Y., Harada, D. and Russell, S. J., Policy invariance under reward transformations: Theory and application to reward shaping, in *Proc. 16th Int. Conf. Machine Learning* (1999), pp. 278–287.

[49] Pasemann, F., Evolving neurocontrollers for balancing an inverted pendulum, *Network-Comp. Neural* **9** (1998) 495–511.

[50] Pavlov, I., *Conditioned Reflexes* (Oxford University Press, Oxford, UK, 1927).

[51] Phon-Amnuaisuk, S., Learning cooperative behaviours in multiagent reinforcement learning, in *Neural Information Processing*, *LNCS* (2009), pp. 570–579.

[52] Porr, B. and Wörgötter, F., Strongly improved stability and faster convergence of temporal sequence learning by using input correlations only, *Neural Comput.* **18** (2006) 1380–1412.

[53] Porr, B. and Wörgötter, F., Fast heterosynaptic learning in a robot food retrieval task inspired by the limbic system, *Biosystems* **89** (2007) 294–299.

[54] Price, B. and Boutilier, C., Accelerating reinforcement learning through implicit imitation, *J. Artif. Intell. Res.* **19** (2003) 569–629.

[55] Rescorla, R. and Solomon, R., Two process learning theory: Relationship between pavlovian conditioning and instrumental learning, *Psychol. Rev.* **88** (1967) 151–182.

[56] Rescorla, R. and Wagner, A., A theory of pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement, in *Classical Conditioning II*: *Current Research and Theory* (1972) 64–99.

[57] Sherstov, A. and Stone, P., Function approximation via tile coding: Automating parameter choice, in *Proc. Symp. Abstraction, Reformulation*, *and Approximation* (*SARA-05*) (2005), pp. 194-205.

[58] Shibata, K., Nishino, T. and Okabe, Y., Active perception and recoginition learning system based on Actor-Q architecture, *Syst. Comput. Jpn.* **33** (2002) 12–22.

[59] Skinner, B., *The Behavior of Organisms*: *An Experimental Analysis* (Appleton Century Croft, New York, 1938).

[60] Smith, S. C. and Herrmann, J. M., Homeokinetic reinforcement learning, in *First IAPR TC3 Workshop*, *PSL 2011*, *LNCS* (2012), pp. 82–91.

[61] Suri, R. E., Bargas, J. and Arbib, M. A., Modeling functions of striatal dopamine modulation in learning and planning, *Neuroscience* **103** (2001) 65–85.

[62] Sutton, R. and Barto, A., *Reinforcement Learning*: *An Introduction* (MIT Press, Cambridge, 1998).

[63] Thorndike, E., Animal intelligence: An experimental study of the associative process in animals, *Psychol. Rev. Monogr. Suppl.* **8** (1898) 68–72.

[64] Tokic, M., Adaptive ϵ-greedy exploration in reinforcement learning based on value differences, in *Proc. KI 2010: Advances in Artificial Intelligence* (2010), pp. 203–210.
[65] Touretzky, D. and Saksida, L., Operant conditioning in skinnerbots, *Adapt. Behav.* **5** (1997) 219–247.
[66] van Dijk, S. G. and Polani, D., Grounding subgoals in information transitions, in *Proc. IEEE Symp. Adaptive Dynamic Programming and Reinforcement Learning* (Paris, France, 2011), pp. 105–111.
[67] Watkins, C. J. C. H., Learning from delayed rewards, Ph.D. thesis, University of Cambridge (1989).
[68] Williams, D. and Williams, H., Auto — maintenance in the pigeon: Sustained pecking despite contingent non-reinforcement, *J. Exp. Anal. Behav.* **12** (1969) 511–520.
[69] Woolley, B. G. and Stanley, K. O., Evolving a single scalable controller for an octopus arm with a variable number of segments, in *Parallel Problem Solving from Nature*, *Lecture Notes in Computer Science* (2010), pp. 270–279.
[70] Wörgötter, F. and Porr, B., Temporal sequence learning, prediction and control — A review of different models and their relation to biological mechanisms, *Neural Comp.* **17** (2005) 245–319.