Reinforcement-Driven Shaping of Sequence Learning in Neural Dynamics

Matthew Luciw[1], Sohrob Kazerounian[1], Yulia Sandamirskaya[2], Gregor Schöner[2] and Jürgen Schmidhuber[1]

1. Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Manno-Lugano, Switzerland
2. Institut für Neuroinformatik at the Universitätstr, Bochum, Germany
{matthew,sohrob,juergen}@idsia.ch,
{yulia.sandamirskaya,gregor.schoener}@ini.ruhr-uni-bochum.de

Abstract. We present here a simulated model of a mobile Kuka Youbot which makes use of Dynamic Field Theory for its underlying perceptual and motor control systems, while learning behavioral sequences through Reinforcement Learning. Although dynamic neural fields have previously been used for robust control in robotics, high-level behavior has generally been pre-programmed by hand. In the present work we extend a recent framework for integrating reinforcement learning and dynamic neural fields, by using the principle of shaping, in order to reduce the search space of the learning agent.

Keywords: Neural Dynamics, Behavioral Organization, Shaping, Reinforcement Learning, Sparse Reward Functions

1 Introduction

In the past few years, there has been a renewed interest in Dynamic Field Theory (DFT) for robust control of robotic agents [32, 28, 25]. In DFT, attractor dynamic based behaviors stably deal with noisy and time-continuous sensory input. making it a desirable candidate for a method of sensory processing and motor control.

To enable an agent to perform high-level, goal-directed action sequences, the behavioral repertoire of an agent can be organized into Elementary Behaviors (EBs). Each EB has an "intention", a stable attractor state persisting during the behavior, and a "condition of satisfaction" (CoS; the desired perceptual outcome of a behavior), a stable attractor state which destabilizes the intention, and therefore sits between EBs. EBs can be chained together to perform action sequences. Although preprogrammed behavioral sequences may suffice in certain environments, a more general autonomous agent should be able to learn new rewarding behavioral sequences, and adapt to different environments, online and in real-time.

In order to show how goal-directed sequences of EBs could be learned from reward, we recently introduced a model [17] which blended Dynamic Neural Fields, and the Reinforcement Learning (RL; [37, 16]) algorithm, SARSA(λ). This system could autonomously learn sequences through random exploration, and the model operated in real-time, continuous environments.

One drawback with our previous model is that, as the number of available behaviors increases, waiting for such an agent to randomly explore action sequences in real-world environments becomes untenable. Building upon our previous framework, we introduce here a study of how the concept of shaping [35, 22], from the field of animal learning can be used in order to speed up training time for robotic and artificial agents operating in our learning framework. Shaping has elsewhere been applied to robot learning [13, 20, 2, 30, 5], but not before within a DFT-based framework. DFT-based EBs provide a robust interface to noisy and continuous environments, RL provides autonomous learning through exploration, and shaping accelerates learning without abandoning our inspiration: a developmental narrative that goes back to Piaget [23].

2 Background

2.1 Dynamic Field Theory

While classical neural network architectures make use of computational units at the level of individual neurons, Dynamic Field Theory (DFT; [32]) is a framework built on Amari dynamics [1], which mathematically describe the continuous-time dynamics of activitions over a field of neurons. The activity of any given Dynamic Neural Field (DNF) is defined over continuous dimensions (e.g., color or space), which characterize the sensorimotor systems and task space of the agent. Fields aggregate neural activity by simulating excitatory inputs, as well as lateral patterns of connectivity, such as local excitatory, and long-range inhibitory connections. As a result of the non-linearities in the DNF's dynamics, and the lateral interactions within the fields, stable localized *peaks* of activation emerge from distributed, noisy, and transient input. These activation peaks represent perceptual objects or motor goals in the DFT framework.

This ability to form and stabilize robust categorical outputs, makes DFT architectures particularly well suited for robotic control systems. Multiple coupled DNFs spanning different perceptual and motor modalities can be composed into complex DFT architectures to organize robot behavior. The building blocks of these architectures are known as Elementary Behaviors.

2.2 Elementary Behaviors and Behavior Chaining

An Elementary Behavior (EB) is an organizational structure in DFT which not only defines the actions associated with a behavior, but also the mechanisms for initiating and terminating that behavior.

The key elements in a DFT EB are the intention and the condition of satisfaction. Both the intention to act and the associated condition of satisfaction are represented by attractor states within dynamic neural fields. Because the amount of time needed to complete an action may vary unpredictably in dynamic and partially unknown environments, the intention to achieve the behavior's goal is maintained as a stable state until completion of the goal is signaled, so it is not necessary to model how long it is expected for the behavior to take. The condition of satisfaction is a neural representation of the sensory conditions that index that an intended action has been completed. The CoS serves two roles: 1. it terminates the associated intention, and 2. as a stable state,

it serves as a breakpoint between behaviors — once satisfaction is achieved, the agent can decide what behavior to do next.

A standard DNF-based EB [27]) consists of a set of DNFs, as well as intention and CoS nodes. While the nodes play a role at the level of inter-behavior dynamics (i.e., switching between behaviors by initiating and terminating a given EB), the DNFs in the EB determine the intra-behavior dynamics (e.g., how the motor system responds in real-time, to changing perceptual stimuli).

In previous work, we have shown how EBs may be chained according to rules of behavioral organization [25, 27], serial order [29, 7, 6], or the value-function of a goal-directed representation [17]. Multiple EBs can be composed into chains [28], where a sequence of behaviors execute one after another, in parallel and/or in response to sensory information [27].

In order to introduce learning into the scheme, adaptive weights can be placed between CoS nodes and intention nodes, representing transitions between a just completed behavior (CoS), and possible next behaviors (intention). These weights serve as values in the RL sense.

2.3 Dynamic Neural SARSA

The recently introduced Dynamic Neural SARSA(λ) [17] model integrates the well known SARSA(λ) model of reinforcement learning [26, 37] with the mathematical language of neural dynamics. The model makes use of EBs for sensorimotor control, while simulating the eligibility traces (λ) of the SARSA model with an Item and Order Working Memory [11].

Although the model successfully learns behavioral sequences from delayed rewards, random exploration with the fully connected search space (any EB can transition to any other EB) can become prohibitively time consuming. Not only does random exploration lead to a combinatorial explosion of the search space with increasing numbers of behaviors, but because each behavior is a continuous real-time action, any given action will require a variable amount of time to terminate before exploration can continue. For such a learning mechanism to work in a more efficient, and time-friendly manner, shaping is used.

2.4 Shaping

Shaping, introduced by B.F. Skinner [35, 22], is well-known in both the psychological and reinforcement learning literature as a method of conditioning. Shaping involves teaching a desired behavior by *successive approximations*, where the teacher or trainer invents and rewards subgoals, which bring the agent's behavior closer to that of the desired behavior. For example, in a classic shaping experiment, Skinner trained a pigeon to strike a wooden ball, by successively rewarding the pigeon turning towards the ball, then stepping towards the ball, then moving within a certain distance of the ball, etc. Skinner described the effect of shaping as "altering the general distribution of behavior", noting "in this way we can build complicated operants which would never appear in the repertoire of the organism otherwise". Critically, one of the defining characteristics in shaping, is successive and shifting *positive* rewards, rather than the use of

negative punishments (which can certainly also be used to "alter the general distribution of behavior").

With respect to artificial agents, RL researchers realized that difficulties could arise from some reward functions, such as those with a single goal state in a large search space, when combined with undirected exploration methods, such as ϵ-greedy or purely random search [38, 37]. Undirected exploration methods rely on random actions, resulting in 1. redundancy in the search due to lack of a memory structure and 2. search bias centralized on the starting position, making it more difficult to discover far away rewards. *Informed*, directed exploration methods (such as *optimistic initialization* or *artificial curiosity* [31]) are more effective in accelerating learning. But the learning speed with directed exploration pales in comparison to guided learning, wherein one knowledgable about how to achieve rewards is able to transfer this knowledge to the agent. Guided learning manifests in RL under various guises and names, some of which are, sometimes (but not always), referred to as shaping. These include chaining [39] or chunking of actions into macro-actions [21], manipulating the reward function to guide the agent [13, 20], and knowledge transfer over tasks [33, 18].

Reinforcement learning is, in theory, an attractive framework for autonomous learning. Autonomous robots present difficulties however: they are slow, and prone to breakage [8], and, of course, they have to operate in real-time. Undirected and even directed exploration presents prohibitive challenges for autonomous robots, especially with larger state spaces. Dorigo and Colombetti [5] introduced the term *robot shaping*, wherein a trainer, providing guidance and support, was found to be greatly effective in speeding up the robot's learning. Various methods have been introduced to allow the teacher to reinforce the robot in a timely and useful manner, such as a reinforcement sensor [4], "good" and "bad" buttons [41], as well as related methods such as Learning from Easy Missions (LEM; [2]).

3 Methods

Robot and Environment. The Kuka Youbot was our experimental robotic platform for our system, implemented in the Webots simulator [40]. The Youbot combines a omnidirectional mobile base (via Mechanum wheels [14]) with a one degree of freedom (DOF) rotating base platform, upon which is a standard three DOF RRR arm [36] with a two pronged gripper, with force feedback. The Youbot provides flexibility to move around untethered on a flat surface, and to reach for and grasp small objects. It is a good "far-ranging" pick and place robot, compared to an arm with an immobile base. The Youbot was enhanced with a RGB and kinect sensor on the front, to detect and localize targets for reaching, and infrared range (IR) sensors around the robot, to detect obstacles. The Youbot is placed in an environment with a few differently colored blocks upon boxes, some obstacles, and a deposit location — the container near the oven. A reward is given when the robot transports a object of a specific color into the container at the deposit location. See Fig. 1. Our implementation used seven different elementary behaviors, which we will discuss further below. First we describe EBs in general in terms of attractor dynamics and neural fields.

Attractor Dynamics EBs. As modeled by Bicho, Mallet, and Schöner [3], each EB involves controlling one or more behavioral variables (e.g., heading direction), which

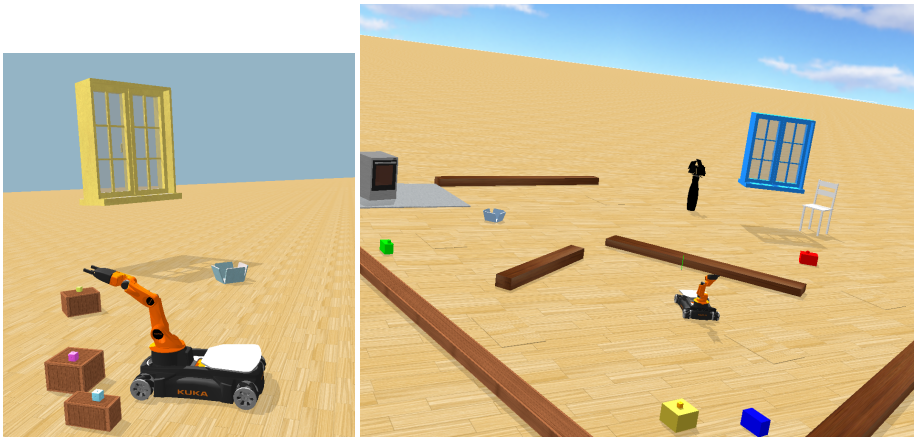

Fig. 1. Left. The YouBot. Right. The environment.

represent the state of the system, particular to that EB. The behavioral variable values are continually mapped to control the robot's effectors. The behavioral dynamics are differential equations updated via attractive and repulsive forcelets. The attractor solutions are the asymptotically stable states, and the achievement of one of these is the goal of the behavior. Example attractive forces are changing the alignment of heading direction until a target is directly in front of the robot, and diminishing the distance between the robot and the target until it is below a threshold. Example repulsive forces are from obstacles, which effectively perturb heading direction and distance so that the robot moves away from them. The definitions and strengths of the attraction, repulsions, and their ranges are hand-selected.

Field-Based EBs. The above attractor dynamics approach is effective but limited. In some cases, such as with a real robot using information from its sensors to represent its world, the necessary variables (such as the target location) are not directly available; instead the robot needs to leverage its sensory information into usable representations. Dynamic neural fields are used to overcome this limitation. Instead of behavioral variables, DNFs use behavioral dimensions (i.e., a selection of sensory input), over which there is a field of activation. Field activations have input from the environment, lateral input (self-excitation, and a localized activation kernel), and EB-specific "top-down" biases (certain dimensions are boosted, to preferably choose a target associated with those dimensions). Stable peaks emerge as output from the activation dynamics, and those peaks are used as *de facto* behavioral variables.

The activation level of a DNF uses the following differential equation (analyzed by Amari [1])

$$\tau \dot{u}(x,t) = -u(x,t) + h + S(x,t) + \int \omega(x-x')\sigma(u(x',t))dx', \qquad (1)$$

over spatial dimension x at each time t, where $h < 0$ is a negative resting level and $S(x,t)$ is the sum of external inputs, for instance from sensors or other DNFs. The

local activation kernel $\omega(\Delta x)$ determines the lateral interaction within the field, e.g., local excitation and long-range inhibition. There is also self-excitation, in the form of σ, an output function, typically a sigmoid. A high output value leads to a stable peak of activation, the unit of representation in DFT. Field-based EBs use three fields, namely perceptual, condition of satisfaction, and motor, and include a perceptual field bias (for target representation), and a CoS field bias (for goal representation).

EBs used.

- **Visual search** is a field-based behavior, which controls the robot's base, and which stabilizes when the target object is central in the robot's vision. The perceptual field's input is a color dimension at all input image columns [29]. The target's color is biased in the perceptual field, such that only that color's appearance in the field can produce an output peak. The perceptual output feeds into the CoS and motor fields (which controls heading direction). The CoS bias is over central image columns, and the CoS only creates a peak if the target is centered. If the target is not visible, a default pseudo-random movement behavior takes over.
- **Approach target** moves the robot towards the target, which must have been found with visual search beforehand (as a precondition [25]) Behavioral variables are heading direction and speed. The IR sensors provide repulsive forces. The CoS is that the distance between the target and robot is small.
- **Orient arm to target** rotates the arm platform until that the angle between the base-gripper vector and the base-target vector becomes nearly zero. This provides an excellent angle of approach for grasping.
- The **reaching** EB uses the Jacobian (which relates joint angle changes to the velocity of the end effector) of the three DOF RRR arm to continually move the point in between the gripper prongs to a point just above the target (a closed form inverse kinematics solution exists for such a manipulator, but does not suit an attractor dynamics framework).
- The **close gripper** EB closes the gripper prongs until the force feedback, resulting from the gripper pressing on the object, surpasses a threshold.
- The **open gripper** EB moves the gripper prongs in the opposite direction until the joint limits are reached.
- **Approach deposit location** has the same dynamics as **approach target**, but uses the deposit location as the target.

Failure State. These behaviors can fail, either due to the lack of a precondition, or something going wrong during execution (like the object not being grasped properly and falling down). We added a **failure** state for such cases. To detect failure, conditions of *dissatisfaction* (CoD) were built into each EB. All behaviors used timer-based CoD's, set to 200 time steps, where each time step in simulation lasted 64 ms. After failure, the robot pose is reset, as is the environment's state (e.g., any displaced objects are placed back in their starting positions).

Reinforcement Learning. We use a particular RL method, called T-learning [10], in which value is associated with transitions between states, instead of states or state-action pairs. Assigning value to transitions is appropriate in our framework because an elementary behavior itself is an attempted transition between two stable states (attractors), and the agent's decision constitutes choosing the next elementary behavior when

it is in one of these stable states. In other words, the agent needs to pick its next ideal transition. T-learning is the simplest RL method for this type of setup. With T-learning and EBs, there is no need to define a separate action set.

Further, T-learning is a more efficient learning method than standard SARSA or Q-learning when some actions are failure-prone, as is the case in many robotic learning environments. In our setup, the rewarding transitions are reinforced when accomplished successfully, but when they fail (e.g., the object slips out of the robot's gripper during an attempted grasp), a different transition is credited — one which goes to a "failure" state. If state-action values were used, and if "grasp" were an action, a failure would de-value this action. T-learning suits the learning of "skilled" behavior, where learning to make difficult transitions is highly rewarding. By difficult, we mean that only a small percentage of possible actions reliably transition to a state associated with high reward. The T-learning agent will continue to try to make that transition after experiencing it just once.

The T-learning update rule is

$$T(s, s') \leftarrow T(s, s') + \alpha \left[r + \gamma T(s', s'') - T(s, s') \right] \tag{2}$$

where s, s', and s'' are three successive stable states (either CoS of an EB, or the failure state), r is a reward, α is the learning rate, and γ is the discount factor. In our implementation, T-learning is combined with eligibility traces, in the same way SARSA becomes SARSA(λ) [37].

4 Experimental Results

In the sort of behavioral chaining task we are considering here, we need to reinforce a *sequence* of behaviors, which led to a reward. This is not a Markovian setup. In the rewarding sequence $A \rightarrow B \rightarrow C$, the reward is delivered after C, but C is only a valuable behavior to select after B and A. If we reinforce the selection of C, the robot will often wrongly select it. One way to deal with this is to use composite states such as AB, BC, or ABC, which causes an explosion in the number of states, but can be tractable if we know the necessary chaining limit. Another way is to use an eligibility trace [19, 15], which is what is done in our implementation.

In our previous implementation, using DN-SARSA(λ), the agent needed to discover the rewarding sequence, through random search, before any reinforcement, i.e., learning of the value node weights, could be done. For n behavior nodes, the chance that a sequence of length m is discovered randomly is $1/(n^m)$, which becomes prohibitively small as the number of EBs and/or the length of the sequence increases. Learning, while theoretically guaranteed, becomes too slow for real world agents.

With shaping, the teacher's input modifies the reward function by providing positive reinforcement for successfully completed intermediate steps. We expect shaping decreases *both* the search time to discover the rewarding sequence (i.e., perform it for the first time), and to *learn* the policy that allows the robot to accomplish the sequence reliably. To test this, we compared two RL setups, in which 100 learning trials were run in each. In both, there were four behaviors, and the rewarding sequence was 5 items long (e.g., $A \rightarrow D \rightarrow C \rightarrow B \rightarrow A$). In one case (with shaping), the agent received

rewards after each step in the sequence (besides the first was completed). In the other (without shaping), the agent only received a reward after the entire sequence was completed. In each case, ϵ-greedy action selection was used, and we tested two *exploration settings* of $\varepsilon = \{0.999, 0.996\}$, where the random action chance starts from $\epsilon = 1$ (100%) and is multiplied by ε after each behavioral transition. In all cases, the learning rate for SARSA $\alpha = 0.1$, the eligibility trace parameter $\lambda = 0.6$, and the discount factor $\gamma = 0.9$.

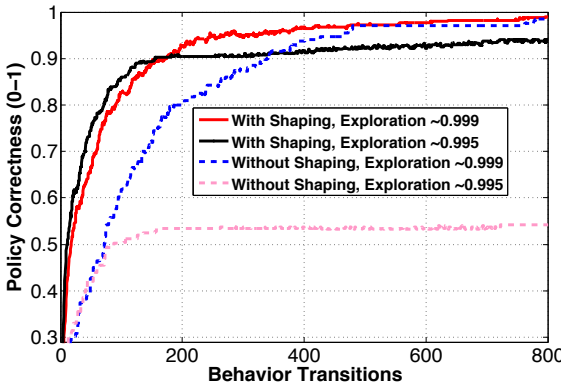

Fig. 2. Shaping improves the speed of policy learning, with an appropriate exploration setting.

In order to measure policy correctness, we measured the deviation of the optimal policy from the policy at every time step. To do this, the maximum valued action of each state was compared with the correct action. If they matched, the agent scored 0.25 points. A perfect policy, where the agent does the best thing in each state, scores 1, and the worst policy scores 0. Mean policy correctness for the four methods at each decision point, or behavior transition point, are shown in Fig. 2. One can see that on average, the two variants with shaping learn much faster. The variants with a quicker decreasing exploration factor are more prone to catastrophic failure runs, where the agent never learns the rewarding sequence. If one can tolerate a chance of failure, the variant with shaping and quicker exploration decrease can learn a bit faster than the other shaping variant. The variant without shaping and quick exploration decrease is very prone to failures.

In Table 1, we can see the average time each variant *first* discovered the rewarding sequence. We also try a few different learning rates, here. The number to the left of the dash in each table cell shows the mean sequence discovery time, while the number to the right of the dash shows the mean time, if all failure runs are removed. In parentheses is how many failure runs there were (out of 100). One can see the shaping variants usually find the sequence first. The variant without shaping and fast exploration decrease fails more often than not, but the runs that don't fail find the sequence very quickly. This however, is simply due to sampling bias (namely, by throwing out all long-run trials that do not complete). The only way this variant can learn the correct sequence is if it gets lucky with the random number generator, and finds the correct sequence early on.

Table 1. Format: a-b (c). (a): Mean number of decisions before the rewarding five-item sequence was first discovered. (b) Each run was stopped after 2000 decisions, and "catastrophic runs" never find the sequence. The second number shows the average without including catastrophic runs. (c): The number of catastrophic runs, out of 100.

		Learning Rate		
		0.1	0.25	0.5
With Shaping	$\varepsilon = 0.999$	401 - 401 (0)	**372** - 372 (0)	400 - 400 (0)
	$\varepsilon = 0.995$	**311** - 259 (3)	391 - 270 (7)	**390** - 287 (6)
Without Shaping	$\varepsilon = 0.999$	597 - 507 (6)	670 - 539 (9)	600 - 511 (6)
	$\varepsilon = 0.995$	1280 - **197** (*60*)	1291 - **182** (*61*)	1233 - **173** (*58*)

We successfully applied the shaping-based RL system to the Youbot, to produce the behavior of depositing a yellow target object in the periwinkle container. A video of the youbot performing the rewarding sequence is at `http://www.idsia.ch/~luciw/videos/youbotreward.avi`. A few behavior transition failures can be seen at `http://www.idsia.ch/~luciw/videos/youbotfail1.avi` (the robot tries to reach an object out of range), and `http://www.idsia.ch/~luciw/videos/youbotfail2.avi` (the robot tries to go to a target it has not located visually, yet).

5 Discussion and Conclusions

Without shaping, the robot had to explore, either randomly, or over every possible sequential path. Although this could be effective in discovering and reinforcing basic organizational constraints between behaviors, the speed of learning the overall goal would be significantly slower. With shaping, the goal was reached faster. Both of these modes could be appropriate in different situations. While a naive robot may need to explore broadly to discover what it can and cannot do at a low level (assuming it is protected from harming itself [8]), it can be desirable to allow a teacher to guide learning for particular tasks. In such scenarios, shaping becomes an efficient and effective method of learning.

A potential drawback of imposing shaping or some other guided learning method on an autonomous agent is that the agent loses some autonomy by being guided. In other words, the agent might not only learn to do what we want it to do, but also how to do it. Skinner's pigeon could not be directly programmed, but our robots can be. What is the utility of learning? Why don't we just program the robot to do what we want?

The same arguments supporting RL over direct programming apply to shaping, except at a lower level of granularity. Shaping, in the form suggested by Skinner's experiments, involves constructing "waypoints", in the reward function, without imposing the exact path to get to each waypoint. Adaptive algorithms can take advantage of unforeseen environmental quirks and find surprising (to the teacher) paths. And some agent-environments are not straightforward to hand-program (i.e., bicycle [24], cart-pole [9]).

Further, shaping does not have to be the only method of learning available. Shaping can be an important piece of the learning repertoire of any autonomous agent that will

interact with teachers/trainers in real time. The robot can use other forms of learning (i.e., curiosity-driven exploration) when it has time to do so, and when it won't hurt itself. In other cases, the teacher can play a very important role.

Future Work. A deficiency of our architecture is that it cannot learn a sequence with multiple instances of the same item in it, such as $A \rightarrow B \rightarrow A \rightarrow C$. This is a known issue of sequence learning with an Item and Order working memory [12], which is what we used for the eligibility trace. Any method where a temporal sequence becomes a spatial pattern via decay will be unable to represent a sequence with repeated elements, without using duplicated but unique elements, sometimes called *rank cells* (e.g., $A-1$, $A-2$ can represent two instances of A [34]).

A simple method to deal with the repeated items problem, in the context of this work, is to ensure that no rewarding sequence includes repeated elements. Of course, this will diminish the sequence space that is searched over. It is sensible that, given such a shortcoming, we should not allow exploration of sequences with repeated items either, but our current system has no such restrictions. We are currently exploring neural dynamics mechanisms for disabling exploration over repeated items. We note this will improve the learning speed of the current system, as the probability of randomly getting the right sequence of m items in n possibilities increases greatly, from $1/(n^m)$ to $1/(n!/(n-m)!)$.

Acknowledgments. This work was funded through the 7th framework of the EU in grant #270247 (NeuralDynamics project).

References

1. Amari, S.: Dynamics of pattern formation in lateral-inhibition type neural fields. Biological Cybernetics **27** (1977) 77–87
2. Asada, M., Noda, S., Tawaratsumida, S., Hosoda, K.: Purposive behavior acquisition for a real robot by vision-based reinforcement learning. In: Recent Advances in Robot Learning. Springer (1996) 163–187
3. Bicho, E., Mallet, P., Schöner, G.: Target representation on an autonomous vehicle with low-level sensors. The International Journal of Robotics Research **19**(5) (2000) 424–447
4. Colombetti, M., Dorigo, M.: Training agents to perform sequential behavior. Adaptive Behavior **2**(3) (1994) 247–275
5. Dorigo, M.: Robot shaping: an experiment in behaviour engineering. The MIT Press (1998)
6. Duran, B., Sandamirskaya, Y.: Neural dynamics of hierarchically organized sequences: a robotic implementation. In: Proceedings of 2012 IEEE-RAS International Conference on Humanoid Robots (Humanoids). (2012)
7. Duran, B., Sandamirskaya, Y., Schöner, G.: A dynamic field architecture for the generation of hierarchically organized sequences. In Villa, A.E., Duch, W., Erdi, P., Masulli, F., Palm, G., eds.: Artificial Neural Networks and Machine Learning – ICANN 2012. Volume 7552 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2012) 25–32
8. Frank, M., Leitner, J., Stollenga, M., Förster, A., Schmidhuber, J.: Curiosity driven reinforcement learning for motion planning on humanoids. Frontiers in neurorobotics **7** (2013)
9. Gomez, F., Miikkulainen, R.: 2-d pole-balancing with recurrent evolutionary networks. In: Proceedings of the International Conference on Artificial Neural Networks, Citeseer (1998) 425–430
10. Graziano, V., Gomez, F.J., Ring, M.B., Schmidhuber, J.: T-learning. CoRR **abs/1201.0292** (2012)

11. Grossberg, S.: Behavioral contrast in short-term memory: Serial binary memory models or parallel continuous memory models? Journal of Mathematical Psychology **3** (1978) 199–219
12. Grossberg, S., Kazerounian, S.: Laminar cortical dynamics of conscious speech perception: Neural model of phonemic restoration using subsequent context in noise. The Journal of the Acoustical Society of America **130**(1) (2011) 440–460
13. Gullapalli, V.: Reinforcement learning and its application to control. PhD thesis, Citeseer (1992)
14. Indiveri, G.: Swedish wheeled omnidirectional mobile robots: kinematics analysis and control. Robotics, IEEE Transactions on **25**(1) (2009) 164–171
15. James, M.R., Singh, S.: Sarsalandmark: an algorithm for learning in pomdps with landmarks. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems (2009) 585–591
16. Kaelbing, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of Artificial Intelligence Research **4** (1996) 237–285
17. Kazerounian, S., Luciw, M., Richter, M., Sandamirskaya, Y.: Autonomous reinforcement of behavioral sequences in neural dynamics. In: International Joint Conference on Neural Networks (IJCNN). (2013)
18. Konidaris, G., Barto, A.: Autonomous shaping: Knowledge transfer in reinforcement learning. In: Proceedings of the 23rd international conference on Machine learning, ACM (2006) 489–496
19. Loch, J., Singh, S.: Using eligibility traces to find the best memoryless policy in partially observable markov decision processes. In: In Proceedings of the Fifteenth International Conference on Machine Learning, Citeseer (1998)
20. Mataric, M.J.: Reward functions for accelerated learning. In: ICML. Volume 94. (1994) 181–189
21. McGovern, A., Sutton, R.S., Fagg, A.H.: Roles of macro-actions in accelerating reinforcement learning. In: Grace Hopper celebration of women in computing. Volume 1317. (1997)
22. Peterson, G.B.: A day of great illumination: Bf skinner's discovery of shaping. Journal of the Experimental Analysis of Behavior **82**(3) (2004) 317–328
23. Piaget, J.: The origins of intelligence in children. International Universities Press, New York (1952)
24. Randlov, J., Alstrom, P.: Learning to drive a bicycle using reinforcement learning and shaping. In: Proceedings of the Fifteenth International Conference on Machine Learning. (1998) 463–471
25. Richter, M., Sandamirskaya, Y., Schöner, G.: A robotic architecture for action selection and behavioral organization inspired by human cognition. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS. (2012)
26. Rummery, G., Niranjan, M.: On-line Q-learning using connectionist systems. University of Cambridge, Department of Engineering (1994)
27. Sandamirskaya, Y., Richter, M., Schöner, G.: A neural-dynamic architecture for behavioral organization of an embodied agent. In: IEEE International Conference on Development and Learning and on Epigenetic Robotics (ICDL EPIROB 2011). (2011)
28. Sandamirskaya, Y., Schöner, G.: Dynamic field theory of sequential action: A model and its implementation on an embodied agent. In Scassellati, B., Deak, G., eds.: International Conference on Development and Learning ICDL'2008. (2008) paper 53, 8 pages
29. Sandamirskaya, Y., Schöner, G.: An embodied account of serial order: How instabilities drive sequence generation. Neural Networks **23**(10) (December 2010) 1164–1179
30. Sasksida, L.M., Raymond, S.M., Touretzky, D.S.: Shaping robot behavior using principles from instrumental conditioning. Robotics and Autonomous Systems **22**(3) (1998) 231–249

31. Schmidhuber, J.: Curious model-building control systems. In: Proceedings of the International Joint Conference on Neural Networks, Singapore. Volume 2., IEEE press (1991) 1458–1463
32. Schöner, G.: Dynamical systems approaches to neural systems and behavior. In Smelser, N.J., Baltes, P.B., eds.: International Encyclopedia of the Social & Behavioral Sciences, Oxford, Pergamon (2002) 10571–10575
33. Selfridge, O.G., Sutton, R.S., Barto, A.G.: Training and tracking in robotics. In: IJCAI, Citeseer (1985) 670–672
34. Silver, M.R., Grossberg, S., Bullock, D., Histed, M.H., Miller, E.K.: A neural model of sequential movement planning and control of eye movements: Item-order-rank working memory and saccade selection by the supplementary eye fields. Neural Networks **26** (2012) 29–58
35. Skinner, B.F.: The behavior of organisms: An experimental analysis. (1938)
36. Spong, M.W., Hutchinson, S., Vidyasagar, M.: Robot modeling and control. John Wiley & Sons New York (2006)
37. Sutton, R., Barto, A.: Reinforcement learning: An introduction. Volume 1. Cambridge Univ Press (1998)
38. Thrun, S.B.: The role of exploration in learning control. Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches. Van Nostrand Reinhold, New York (1992)
39. Touretzky, D.S., Saksida, L.M.: Operant conditioning in skinnerbots. Adaptive Behavior **5**(3-4) (1997) 219–247
40. Webots: http://www.cyberbotics.com Commercial Mobile Robot Simulation Software.
41. Weng, J.: Developmental robotics: Theory and experiments. International Journal of Humanoid Robotics **1**(02) (2004) 199–236