Abstract—We introduce a dynamic neural algorithm for learning a behavioral sequence from possibly delayed rewards. The algorithm, inspired by prior Dynamic Field Theory models of behavioral sequence representation, is called Dynamic Neural (DN) SARSA(\(\lambda\)), and is grounded in both neuronal dynamics and classical reinforcement learning. DN-SARSA(\(\lambda\)) is implemented on both a simulated and real mobile robot performing a search task for a specific sequence of color finding behaviors.

I. INTRODUCTION

Computational approaches to reinforcement learning (RL) often formalize the learning problem in terms of discrete state and action spaces, with a learning agent that operates in discrete time [1]. The problem of how these discrete representations emerge from sensory-motor representations, which are continuous in time and in space, is often not addressed in the RL literature. On the other hand, some RL models in neuroscience include the continuous neural representations of states and actions, but they do not address the problem of learning sequences of behaviors through reinforcement, as well as how these sequences may be generated using real sensors and motors [2], [3].

Here, we present a neural-dynamic model that implements an RL agent, which is able to acquire action sequences based on a reward signal and uses the state and action representations that may be continuously linked to raw perceptual inputs and motor dynamics. In the neural-dynamic RL architecture, the behavioral decisions are modeled as instabilities in the dynamics of neural fields which are continuous in time and are graded in space. These instabilities demarcate transitions between stable states that represent the agent’s actions, as they unfold continuously in physical time and environment. As stable states emerge from the continuous dynamics, they form the basis for building neural-dynamic representations of previously selected pairs of states and actions, their eligibility traces, and value function of the reinforcement learner.

The model uses the neural-dynamic framework of Dynamic Field Theory (DFT) [4] to represent the behaviors of the agent that form the state-action space, on which learning operates. The well-known RL algorithm SARSA is used to implement the reinforcement learning of action sequences over these representations. We provide a method for autonomously discretizing behaviors occurring in a real-time continuous neural-dynamic framework that enables RL in continuous sensory-motor processes. We implement the model, which we call Dynamic Neural (DN) SARSA(\(\lambda\)), in a simple color sequence learning scenario and demonstrate its functioning on a real robot.

II. BACKGROUND

A. Dynamic Field Theory

Our model is based on Dynamic Field Theory (DFT) [5], a mathematical framework for cognitive processes. Within DFT, dynamic neural fields (DNFs) are used to represent activation distributions of neural populations. The activation is defined over graded metric dimensions (e.g., color or space) relevant to the task and develops in continuous time based on Amari dynamics [18]. Stable peaks of activation form as a result of supra-threshold activation and lateral interactions within a field. Due to its process model nature, DFT architectures are able to deal with continuous time and real world environments and are thus well suited for robotic control systems.

B. Previous Mechanisms of Learning in DFT

The basic learning mechanism in DFT is a memory trace of the positive activation of a DNF. This mechanism has been used to model long-term memory with respect to task space [6], [7], the motor memory of previous movements [8], [9], to encode invariant features [10], and to represent locations of objects [11]. In these models, learning is achieved by the dynamics of the memory trace’s build-up and decay. Memory traces of multi-dimensional DNFs implement associative learning between different modalities. Such associations may be used to encode serial order of actions [12] or associations of features and their locations in space [11], [13].

In previous work, sequence learning in DFT amounted to storing memory traces of an observed sequence of behaviors. In this paper, the sequence of behaviors is discovered autonomously based on a delayed and non-specific reward signal.

C. Reinforcement Learning

A general statement of the RL problem, following Sutton and Barto (1998; [1]) assumes an agent which interacts with its environment. At any moment in time \((t)\), the agent experiences the state of the environment \((s_t)\), and on that basis makes a decision about which action to take next \((a_t)\). This decision process is determined by the agent’s policy \((\pi(s,a))\) which maps state to action. The goal then, is that in the course of exploring its environment and receiving rewards at various points, the policy should be updated so that action selection is more likely to result in reward. Some common methods
learn the optimal policy by learning the *value function* (VF). The value of a state-action is (formally) the expected future cumulative discounted reward if the agent takes that action in that state and follows its policy thereafter. Policy iteration alternates between learning the VF of a policy and improving the policy (by selecting the value maximizing action).

In many RL formulations, the environment is structured such that it has the property of being a Markov Decision Process (MDP). Informally, this means that the response of the environment depends solely on the state and action in that moment, independent of the history of states or actions prior. An example would be a game of chess, where the next board configuration depends only on the current configuration and the selected action, rather than the sequence of moves which lead to that configuration. However, in many environments this property does not hold. When response of the environment depends not just on the current state and action, the environment is said to be a partially observable MDP (POMDP).

Lastly, when the problem operates in continuous-time, such that actions are no longer operations that occur in discrete time-steps, the environment is said to be semi-MDP ([1], pg. 276). This occurs in our tested environment, and many real-world environments, in which the action performed requires variable duration in order to complete.

At the heart of traditional RL for MDPs is the idea of Temporal-Difference (TD) learning. TD learning is model free, and updates the value of a particular state (or state/action pair), based on the value of subsequent state(s) (or state/action pairs). The SARSA algorithm is an on-policy method which makes use of TD learning to update state-action values. SARSA($\lambda$) extends that work by introducing the concept of an eligibility trace, which updates not just concurrent state-action values, but the history of state-action values over the course of a trial. That is, if we denote our TD-error at any given time $t$ as $\delta_t$, state-action values which occurred $t$ timesteps back, are updated by a factor of $\gamma^t \delta_t$. The use of eligibility traces has not only been shown to speed up learning, but also been shown to help overcome the problem of learning in POMDPs [14]. For detailed descriptions of these concepts, we refer the reader to [1].

### D. RL and Computational Neuroscience

Since we know that humans and animals learn in real-time, dynamic environments, it makes sense to consider views of RL taken in computational neuroscience. This approach has largely focused on modeling TD learning, as there is accumulating neurophysiological evidence that midbrain dopaminergic neurons encode a form TD-error [2]. Indeed, a number of models have been able to model low-level aspects of reinforcement learning, including sequence production in Basal Ganglia [3], foraging behavior in bees [15], planned and reactive saccades [16]. However, while these models explain an impressive array of physiological data regarding RL, they too make simplified assumptions about the nature of the environments they model. Moreover, they often fail to show how complex behavioral skills can be learned and in most cases do not account for how behavior is generated in continuous time based on realistic sensory information and tied into actual motor systems. It appears, therefore, that the neuroscience approach to RL may also be insufficient for enabling artificial agents and robots to learn in complex environments.

As noted by Kawato and Sanejima [17], there are three primary problems facing neural models of RL. First, standard TD algorithms learn too slowly to be considered realistic methods of learning, either in animals or in robots. Second, the exact mechanisms by which TD-errors are computed by neural circuits remain elusive. Third, neural models of RL fail to explain complex behavioral learning which incorporate cerebral cortex and cerebellum. It is therefore increasingly clear that theories of learning will have to integrate both algorithmic and neuroscience traditions, in order to describe (and model) how learning scales up to complex, real-time and dynamic environments.

The goal of DN-SARSA($\lambda$) is to provide a framework which can begin to address these difficulties, by showing how computational enhancements to learning, such as eligibility traces, can be realized in neural circuits; to propose a mechanism by which TD-errors with eligibility traces can be computed, while maintaining the Bellman consistency; and to show how neural reinforcement learning algorithms can interact with sensory cortices, all of which operate in real-time, on real inputs.

### III. THE DN-SARSA($\lambda$) ARCHITECTURE

#### A. Overview

The DN-SARSA($\lambda$) model consists of a neural-dynamic architecture for generation of behavioral sequences as well as a neural-dynamic reinforcement learner. A number of coupled dynamic neural fields (DNFs) [18] and neural nodes form a representation of the elementary behaviors (EBs) of the agent's behavioral repertory. Each EB has a DNF representation of the intention and of the condition-of-satisfaction (CoS) of the respective behavior. Both these representations are graded in space and continuous in time attractor dynamics, which may be coupled to perceptual and motor systems of a robotic agent. The intention DNF interacts with bottom-up sensory inputs to drive low-level motor commands. Activation of the CoS DNF indicates that the currently active behavior has completed [19].

For the reinforcement learner, an active CoS field represents the *state*, in which the agent decides which *action* to activate next (represented by the intention DNF of the next EB). A state/action DNF of the reinforcement learner receives inputs from CoS fields and the intention fields of the EBs and builds a peak of positive activation in each transition phase between EBs, when the CoS field of the previous EB is still active and the intention field of the next EB is already activated.

The positive activation in the state/action DNF ultimately serves as input to an Item and Order working memory system [20], [21]. Activity in this system represents an eligibility trace, since the more recently occurring state/action transitions result in higher levels of activity than those state/action transitions having occurred further in the past. The eligibility
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 Gaussian-shaped kernel \( \omega(\Delta x) \) determines the lateral interaction within the field. For supra-threshold activation, this interaction leads to stable peaks of activation, the unit of representation in DFT.

2) Elementary Behaviors: In order to represent actions (e.g., "move to red object") in a real-world environment and in continuous time, we use a DFT based model of an elementary behavior [22]. An EB consists of two dynamical structures: a representation of the intention (e.g., move toward red object) and of the condition of satisfaction (e.g., the agent is at the red object). At every point in time, the CoS DNF matches the intention with the current sensory input. Upon a successful match, the CoS signals the completion of the EB and deactivates its intention. The structure of EBs enables segmentation of a continuous behavioral flow into discrete intentional (goal-directed) actions.

To represent the above, we’ve used coupled intention and CoS nodes, linked to perceptual and CoS fields: An example of a perceptual field is one which takes camera input, and transforms it so that the y-axis represents maximum hue, and the x-axis is pixel column [19]. The corresponding CoS field, defined over the same axes as the perceptual field, serves as input to the CoS nodes. Intention nodes provide top-down biases to the perceptual and CoS fields, and these biases effectively define the behaviors. An intention node of a particular EB (e.g., “find yellow”) will bias the appropriate hue in the perceptual field and the appropriate area (e.g., the center) of the CoS field. Bottom-up input from the CoS field to an EB’s CoS node allow the node to become active in response to the stimuli which define when the behavior has been completed [19].

Superposition of the perceptual field and the preshape from intention nodes results in regions of super-threshold activity, which then drive low-level motor commands via the motor field, e.g., setting an equilibrium point for a muscle or an angular velocity for the wheels of a mobile robot. An example motor field is a simple 1D space representing heading direction. As the agent performs an action, environmental stimuli such as visual input from cameras, or position information from motor encoders, change continuously in time, resulting in changes in the pattern of activity across the perceptual field.

The intention nodes balance self-excitation, inhibition from its own CoS node, and excitation from its value node (value nodes are explained later). The parameters are tuned so that, when no intention node is above threshold (sigmoidal \( f \) is near zero for all) a winner-take-all behavior results. Otherwise, a single intention node stays “on” (high \( f \)) due to self-excitation and suppression of the others. The equation for each intention node’s activity is given by:

\[
\tau_{int} \dot{i}_i^{int} = -d_i^{int} + h_i^{int} + c_i^{int} f_S(d_i^{int}) + c_v^{int} d_i^{val} - c_i^{int} \sum_{k \neq i} f_S(d_k^{int}) - c_v^{int} f_S(d_i^{val})
\]  

\[(2)\]

B. Sequence Generation Dynamics

1) Dynamic Neural Fields: The activation level of DNFs develops in time based on the following differential equation, as analyzed by [18]

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

\[ (1)\]
The CoS nodes signal when a behavior has been completed, on the basis of bottom-up perceptual input. The equation for each CoS node is given by:

$$
\tau \frac{\cos d_{i}^\text{cos}}{d_{i}^\text{cos}} = -d_{i}^\text{cos} + h_{i}^\text{cos} + c_{+}^\text{cos} f_S(d_{i}^\text{cos})
- c_{\text{int}} \sum_{k \neq i} f(d_{k}^\text{int}) + c_{\text{input}} \sum_{j} f(U_{i,j})
$$

(3)

Activities of both nodes, \((d_{i}^\text{int,cos})\), in the absence of excitatory or inhibitory inputs are driven by a resting level, \(h_{i}^\text{int,cos}\) as well as a passive decay term, \(-d_{i}^\text{int,cos}\) which drives the node’s activity back towards a resting equilibrium. Self-excitatory feedback \((c_{+}^\text{int,cos} f_S(d_{i}^\text{int,cos}))\) stabilizes activity of a node if an external input pushes it through the activation threshold. Lateral inhibition \((-c_{\text{int}} \sum_{k \neq i} f_S(d_{k}^\text{int}))\) among intention nodes causes these nodes to compete in a winner-take-all fashion, such that only a single intention node can remain on, while suppressing others. This competition is biased by nodes which encode learned values via the term \(c_{\text{input}}\). Unlike the intention nodes, the CoS nodes receive bottom-up inputs from the perceptual field \(c_{\text{input}} \sum_{j} f(U_{i,j})\) which excite a CoS node when environmental conditions match the expected context which defines that a behavior is completed. Once a behavior is completed, the CoS node of the given behavior will become active, and shut down the active intention node by inhibitory inputs \(-c_{\text{int}} f_S(d_{i}^\text{cos})\).

In our simulation, we set the parameters in these equations as \(\tau_{\text{int}} = \tau_{\text{cos}} = 3\), and \(h_{i}^\text{int} = h_{i}^\text{cos} = 5\). The inhibitory coefficients were set to \(c_{\text{int}} = 10\) and \(c_{\text{cos}} = 5\) and \(-c_{\text{cos}} = 2\), while the excitatory coefficients were set as \(c_{+}^\text{int} = 10\) and \(c_{\text{cos}}^\text{int} = 20\).

The sigmoid function \(f_S\) ensures that output activations are bounded between 0 and 1, and is given by:

$$
\frac{1 + \beta(x - \mu)}{2(1 + \beta|x - \mu|)}
$$

(4)

Because of winner-take-all (WTA) competition between intention nodes of the EBs, only a single behavior can be selected and active at any given time. This competition is driven either by 1. endogenous random activity (during exploration), or 2. by long-term memory representations of values (during exploitation). These values, stored in weights \(W_{i,j}\), can be read out into value nodes. In the absence of randomized exploration, the value weights specify a chain of behaviors. They cause one behavior to reliably follow another. Ideally, the chain of behaviors will serve to maximize the agents expected future reward.

The activity of the value nodes is computed as:

$$
\tau \frac{\text{val} d_{i}^\text{val}(t)}{d_{i}^\text{val}} = \sum_{j} f(d_{j}^\text{cos}) W_{i,j}.
$$

(5)

Afterwards they are divisively normalized to sum to one.

The value nodes, intention nodes, CoS nodes, perceptual and motor fields work together to produce a sequence of elementary behaviors. In the next subsection, we discuss the RL part, the goal of which is to tune the values.

C. Reinforcement Learner

The second major component of DN-SARSA(\(\lambda\)) is the reinforcement learner. An initial requirement of an RL system is a representation of states and actions.

1) State-Action Representations: In DN-SARSA(\(\lambda\)) a state/action field is a set of discrete nodes organized in a matrix, wherein each row receives input from one of the intention nodes, and each column receives inputs from one of the CoS nodes of the available EBs. The sites in the state/action field are excited in response to coincident activations of CoS and intention nodes, which happens only in a transition phase between two EBs. By detecting transitions in this manner, the states in the RL sense are defined by the CoS nodes (i.e., which behavior the agent has just finished), and the actions are defined by the intention nodes (i.e., what behavior the agent selects next).

The SA cells \((I_{i,j})\) are not implemented as differential equations, but rather assume steady state dynamics, and are defined by:

$$
I_{i,j} = \left(\sum_{k \neq i} \sum_{j \neq i} d_{i}^\text{int} d_{i}^\text{cos}\right) f_H \left(f_S(d_{i}^\text{int}) f_S(d_{j}^\text{cos})\right)
$$

(6)

2) Transient Pulse (TP)-Cells: The activity within the state/action field excite another field of nodes known as transient pulse cells [23]. Each node in this field is modeled as a coupled circuit composed of an excitatory and inhibitory TP cell \((TP^+ and TP^-\) respectively). The activities of each of the \(TP^+\) cells in these circuits behave as onset and offset detectors for their respective state/action nodes, by producing a transient excitatory pulse in response to the onset of input from the state/action field, and a transient inhibitory (negative) pulse in response to the offset of that activity.

The behavior of the field of coupled excitatory \((TP^+_{i,j})\) and inhibitory \((TP^-_{i,j})\) cells is given by:

$$
\tau_{TP} \dot{P}_{i,j}^+ = (-TP^+_{i,j} + I_{i,j} - TP^-_{i,j})
$$

(7)

$$
\tau_{TP} \dot{P}_{i,j}^- = (-TP^-_{i,j} + I_{i,j})
$$

(8)

Both the excitatory and inhibitory cells contain a passive decay term \((-TP^+_{i,j} and -TP^-_{i,j})\), as well as excitatory input from their corresponding state/action cells, \(I_{i,j}\). In addition, the \(TP^+_{i,j}\) receive inhibition from their corresponding inhibitory cell, \(TP^-_{i,j}\). For each intention, \(i\), and each CoS, \(j\), both cells \((TP^+_{i,j} and TP^-_{i,j})\) are initially at rest. When the input, \(I_{i,j}\), from the state/action field turns on, both cells integrate activity at a rate proportional to this input. However, whereas the \(TP^-_{i,j}\) cell integrates activity until it reaches equilibrium (while input remains on, equilibrium is reached at the value of the input), the \(TP^+_{i,j}\) cell will begin to decrease in activity as \(TP^+_{i,j}\) increases. In fact, it is easy to see that at equilibrium, \(TP^+_{i,j} = I_{i,j} - TP^-_{i,j}\), which will therefore approach zero. Once
input shuts off, $TP_{ij}^+$ is approximately 0, whereas $TP_{ij}^-$ is approximately equal to the input strength. As a result, $TP_{ij}^+$ will experience an initial burst of inhibition, until both $TP_{ij}^+$ and $TP_{ij}^-$ then relax back to rest at 0. In both equations, the parameter $\tau = 1/2$.

In DN-SARSA, the onset and offset detection capabilities of TP cells have multiple uses. Firstly, because they exhibit a fixed-width (in time) pulse of activation, they allow buffering of inputs to the eligibility trace layer, in order to prevent persistent inputs to those cells. Secondly, as consequence of the fact that they detect onsets and offsets of inputs, they can serve as the mechanism by which calculation $Q(s', a') - Q(s, a)$ is calculated. That is, if inputs occur in back-to-back fashion, such a mechanism results in the positive activation of TP cells corresponding to the currently active state/action pair $(s', a')$, while simultaneously producing negative activation of the previous state action pair $(s, a)$.

Activity from the $TP^+$ cells serves as input to a neural structure, wherein eligibility traces for the history of the activated state/action pairs is maintained, as described next.

3) Eligibility Trace: Since the eligibility trace in RL [1] may be interpreted as a form of a working memory, we simulate the eligibility trace (ET) field as an Item and Order working memory, which has been used to model a range of behavioral and psychological data regarding working memory, speech perception, and unsupervised sequence learning [20], [21]. Item and Order working memories encode the order of a sequence of presented items by the relative levels of activation across those items. In DN-SARSA($\lambda$), more recently occurring state/action transitions result in higher levels of activity in the ET field than those state/action transitions having occurred further in the past. This property emerges naturally due to a ubiquitous neural architecture, known as a recurrent on-center, off-surround network, whose cells obey shunting dynamics. This structure ensures that the summed total activity is bounded, and that shunting dynamics lead to divisive normalization, which causes individual cell activities to be reduced by constant ratio factors upon presentation of new items. For a more technical analysis, see [24]. Because of the recurrent on-center, off-surround structure, cell activities can reach sustained equilibrium values in the absence of inputs. Further, because the inputs to this field are brief duration pulses corresponding to the onsets of inputs from state/action representations, the activity pattern across this field reaches equilibrium, and is no longer altered regardless of how long the state/action cell itself remains active. Taken together, these processing capabilities give rise to a system which can sustain a fixed activation level as variable length actions are undertaken, and whose activities self-stabilize in periods between, as well as during, subsequent actions.

For a working memory cell which encodes the state/action pair indexed by $(s, j)$, its activity $u_{ij}$ is given by:

$$\tau^u u_{ij} = (1 - u_{ij})\left(\alpha f_P(TP_{ij}^+) + \beta u_{ij}\right) - u_{ij}\left(\alpha \sum_{k,l \neq i,j} f_P(TP_{kl}^-) + \beta \sum_{k,l \neq i,j} u_{kl}\right)$$

The cell’s activity is bounded below by 0, and bounded above by 1 due to the excitatory shunting term $(1 - u_{ij})$, which prevents the inputs from having any effect once $u_{ij} = 1$, and the inhibitory shunting term $-u_{ij}$ which prevents the inhibitory inputs from having any effect once $-u_{ij} = 0$. Inputs from state/action pairs $(I_{ij})$ are pulse inputs resulting from joint activations in CoS and Intention nodes across the EBs. There are also on-center $(\beta u_{ij})$ and off-surround $(\beta \sum_{k,l \neq i,j} u_{kl})$, which, when coupled with shunting dynamics, give rise to the Item and Order properties discussed above. The parameters are set as $\alpha = 1.1$ and $\beta = .8$.

4) Value Opposition Field: The pattern of activity which unfolds across the eligibility trace field excites a value opposition (VO) field, which prepares the calculation of the TD-error. In the VO field, the representations of the currently active state/action pair (with value $Q(s', a')$) and the negative of the previously active state/action pair (with value $Q(s, a)$) become active. This results from the onset/offset detections of state/action pairs by the TP cells in the following way. When the state/action pair $(s', a')$ is selected to be performed, it’s corresponding $TP^+$ cell emits a pulse of activity. At the same time, the previous, just finished state/action pair, $(s, a)$, has a $TP^+$ cell emitting a negative pulse of activity, since its corresponding state/action representation is the most recent one to have turned off. All other $TP^+$ cell activities remain zero. Consequently, the onset / offset detectors simultaneously exhibit excitatory activation in the currently active state/action pair, with inhibitory activation in the previously active state/action pair. These TP-cell activations gate inputs from the eligibility trace field to the VO field. These inputs are also weighted by LTM traces which represent the Q-Values. Together, these multiplicative inputs ensure that the activity in the VO field represent $Q(s', a')$, and $-Q(s, a)$.

Activity in the Value Opposition field follows the dynamics:

$$\tau^O O_{ij} = (-O_{ij} + \gamma f_H(u_{ij})W_{uu} f_H(TP_{ij}^+)) - f_H(u_{ij}) W_{uu} f_H(-TP_{ij}^+),$$

where the function $f_H(u)$ is the Heaviside function. Because the only excitatory $TP^+$ cell activity corresponds to the presently active state action pair, $(s', a')$, and the only inhibitory $TP^+$ activity corresponds to the previously active state action pair, $(s, a)$ at equilibrium gives $O_{ij} = (\gamma W_{(s', a')} - W_{(s, a)})$.

where the weights correspond to our learned Q-values, and the indices i,j have been replaced by the presently and previously active state action pairs. Our parameter $\tau^O = 1/10$, and $\gamma = .8$.

5) TD-Error: The TD-error is calculated in part by a value cell that receives excitatory inputs from all cells in the VO field. This ultimately results in a cell whose activity computes
the difference between the stored LTM values for the currently and previously active State/Action pairs.

The value cell activity is given by:

$$\tau^v \dot{v} = (-v + \sum O_{ij})$$ (11)

Because the LTM weights $W_{vu}$ ultimately come to encode our desired Q-values, the value cell at equilibrium calculates, $v = \sum_{ij} O \approx \gamma W_{Q(s',a')} - W_{Q(s,a)}$. The value stored here then modulates our learning law along with incoming rewards.

6) LTM Weights (Q-Values): The update rule for the weight values of connections between the state/action pairs essential mirror the form of the update equation in SARSA. In particular, the Q-values (that is $Q(s', a')$ and $Q(s, a)$ representing the Q-values of the current and previous s/a pairs) are values of the weights, and the working memory based eligibility trace values correspond to the SARSA eligibility trace values. The weight update equation is Eq. 12,

$$\dot{W}_{ij} = \alpha (1 - SA_{ij})[r + v]u_{ij}$$ (12)

The summed activity across the VO field, plus any external reward present, modulate the weights storing Q-values, as do the eligibility traces which are the pre-synaptic cells to these weights.

---

Fig. 2. Illustration of how our continuous time process model DN-SARSA(\(\lambda\)) converts sensory-motor representations to discrete-like events. Left: the activation of intention and condition of satisfaction nodes during a short chunk of time during the robot’s exploration phase. Left bottom: the intention node output, indicating the behavioral sequence. Not all behaviors take the same amount of time. Right: after learning. The optimal sequence was learned and has stabilized.

IV. EXPERIMENTAL RESULTS

A. Environment and Behaviors

The model is tested on a robotic vehicle simulated in the Webots simulator, performing a search for rewarding sequences of colored blocks, as illustrated in Fig. 3(a). The E-Puck robot is surrounded by 16 blocks of four different colors (red(R), green(G), blue(B), yellow(Y)), which are picked up by the robot’s camera and are represented as localized color-space distributions in the perceptual DNF. The robot “finds” a particular color, as determined by the currently active intention node, by rotating on the spot so that an object of the given color falls onto the center of the image of the vehicle’s camera. Once centered, activation in the CoS node of the particular EB initiates a new EB to be performed (i.e., a new color to be searched for). If the robot finds the correct five-item sequence $G \rightarrow B \rightarrow Y \rightarrow R \rightarrow G$, a positive reward is provided for a few time steps.

Note that this is a POMDP, since our agent’s state encodes the previously completed behavior only. In our environment, the optimal policy is not representable given just the observable state. If we use TD(0), for example, the horizon will be too short — if $R \rightarrow G \rightarrow Y \rightarrow B \rightarrow R$ is uncovered and rewarded, the model will first boost values from $B \rightarrow R$, and will next boost values of any of the three $R \rightarrow B, G \rightarrow B,$
Y \rightarrow B$, but there will be no feedback so that only the correct one could be learned. Memory of the last three behaviors is needed for the true state. Due to the eligibility trace, D Negro-Ramírez et al. 
SARSA(λ) can learn the sequence success fully. It is known that eligibility traces are not a complete solution to POMDPs, but eligibility traces can lead to good or even optimal POMDP solutions in some cases.

B. Setup of the Model

Initially, the value-encoding weights of the reinforcement learner are set to zero. Ultimately, the goal for the robot is that it discovers and learns the target sequence by reinforcement learning. We use a random exploration strategy during the first 50,000 time steps in which noise is added to the weights. This causes the robot randomly select EBs for approximately 300 orientation behaviors that occur during this period. One could imagine future work using more sophisticated exploration methods [25]. After 50,000 steps, the noise is removed and the robot operates in exploitation mode, consistently executing what it estimates to be its most valuable next behavior while continuing to learn.

C. Results

Please see Fig. 2. This illustrates how temporally discrete events emerge from continuous time activation dynamics in the elementary behaviors. These events arise from instabilities in the neural dynamics triggered by CoS onsets. The left column illustrates the irregular activation of EBs during exploration, while the right column shows the consistent sequence of activated EBs in the exploitation phase.

Fig. 3(b)-(e) shows results in terms of the robot’s learning performance. In all trials in which the robot uncovered the rewarding sequence in exploration mode, it was able to eventually execute the optimal policy in exploitation mode.

In some trials, the optimal policy was attained only in the exploitation phase, which showed that it is useful to maintain learning both during exploration and exploitation. Learning in the exploitation phase consists primarily of unlearning incorrect “shortcuts” inherited from the exploration phase. This occurs, for example, when the robot finds the sequence, and correctly values the transition from $R \rightarrow G$ the most, but incorrectly also values the transition from any other color than $Y$ to $R$. During exploitation the robot realizes that shortcuts do not lead to reward (by executing them and not receiving any reward). Their values are diminished until the true rewarding sequence remains.

Fig. 3(c) shows the time at which the sequence was first uncovered. Fig 3(e) illustrates the reward from one run, in which the robot finds the target sequence a first time after about 30,000 steps, finds it again (by luck). When the system enters exploitation mode it starts maximizing reward by doing the correct thing over and over again until the simulation ends. Fig. 3(d) shows the averaged TD-error, illustrating that the neural system learns to predict discounted future reward. The detection of reward acts as an instability for the reinforcement learner, and the learning mechanism is simply a constant drive towards stability.

D. Transfer to Real Robot

To show that our system can deal with real sensory information and real motor system, we transferred a set of weights learned from a successful run of simulation to a real E-puck (see Fig. 4). A video of the robot successfully moving through two iterations of the sequence is at http://www.idsia.ch/~luciw/videos/DFTBot.mp4.

In the video, the top row shows the sensorimotor process: from sensory input to the perceptual field and to the motor field. One can see the different colors that are detected along
the hue dimension (Y-axis of perception), and how priming from the different intention nodes causes selection of one color and execution of the corresponding behavior. Observe that the system is robust against perceptual noise and fluctuation in the visual channel (e.g., changing lighting conditions, shades, mismatch between the robotic and the simulated camera). The activities of the intention and CoS nodes in the bottom row show the behavioral switching dynamics. The CoS field is also shown here, which illustrates the link from perception to behavior completion. Finally, the learned value weight matrix is shown, where white indicates a high value, with CoS (state) on the y-axis and intention (action) on the x-axis. Note that it encodes the rewarding sequence.

The successful transfer onto a real robotic system shows that the DN-SARSA(\(\lambda\)) reinforcement learner brings about a representation that is capable of producing behavior in the physical robot based on continuous (raw) visual input and physical motors, driven by continuous-time dynamics.

![Fig. 4. The E-puck in its environment, surrounded by the colored objects of different sizes and shapes. The "thought bubble" shows the rewarding sequence of colors.](image)

V. CONCLUSION

The DN-SARSA(\(\lambda\)) model provides a framework which shows how computational learning algorithms can be incorporated into a continuous neural-dynamical model. This enables autonomous learning and acting in continuous and dynamic environments, a challenge that is easily overlooked when formalizing the learning problem in discretized spaces without accounting for their coupling to sensory-motor dynamics.

ACKNOWLEDGMENT

We’d like to thank Alexander Förster for mounting the camera on the E-puck. The authors gratefully acknowledge the financial support of the European Union Seventh Framework Program FP7-ICT-2009-6 under Grant Agreement no. 270247 – NeuralDynamics.

REFERENCES


