Self-Organization of Action Hierarchy and Compositionality by Reinforcement Learning with Recurrent Neural Networks

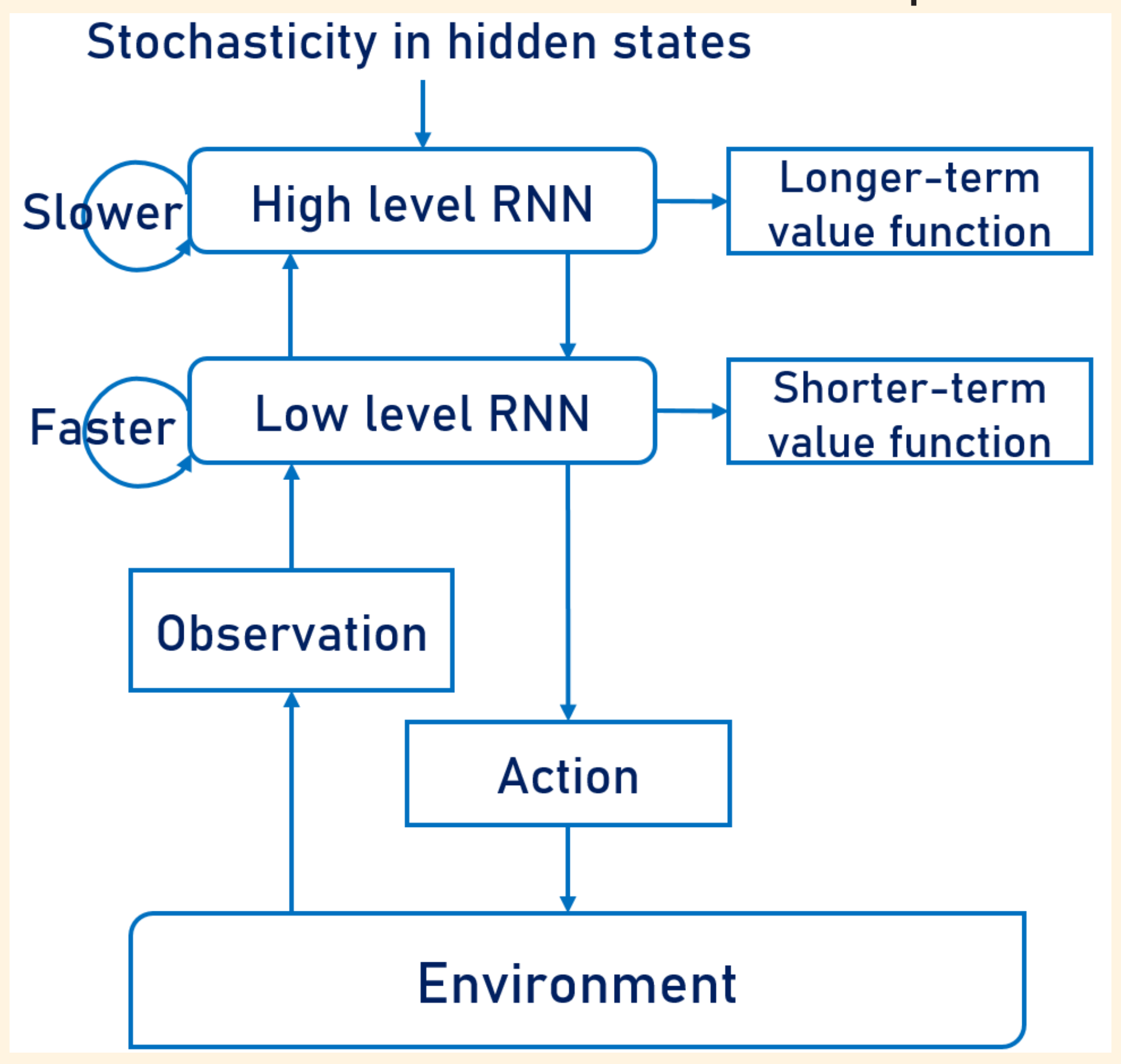
[1]Cognitive Neurorobotics Research Unit
[2]Neural Computation Unit
Okinawa Institute of Science and Technology, Japan

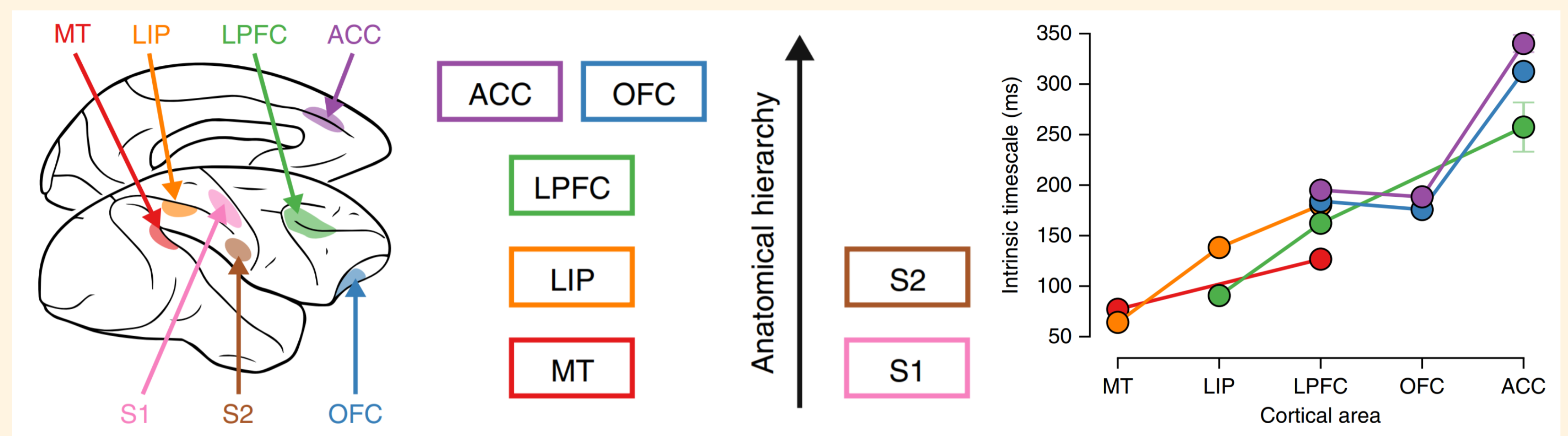
Dongqi Han[1], Kenji Doya[2], Jun Tani[1]
dongqi.han@oist.jp

Introduction

The proposed RL framework: **ReMASTER**

Recurrent Multi-level Actor-critic with STochasctic Experience Reply


Stochasticity in hidden states
Slower
High level RNN
Longer-term value function
Faster
Low level RNN
Shorter-term value function
Observation
Action
Environment



A Hierarchy of Timescales in Primate Cortex. Murray, John D., et al. "A hierarchy of intrinsic timescales across primate cortex." Nature neuroscience 17.12 (2014): 1661.

We are interested in studying the nerual mechnisms regarding the following questions:

- How action primitives (options) that are useful for hierarchical control autonomously develop by RL?
- How the action primitives learned in a task can be re-used to facilitate learning novel tasks?

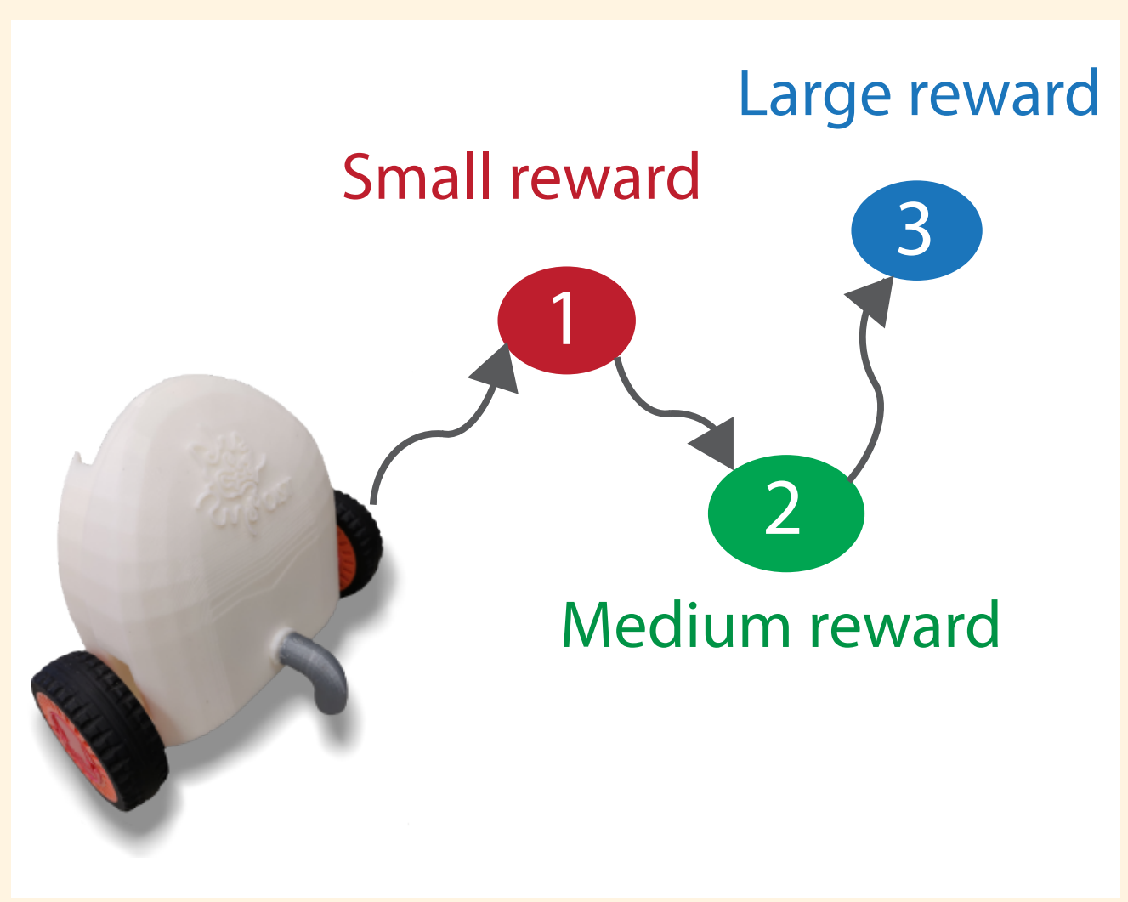
We seek the answers from the biological neural systems:

- The cortex is full of recurrently connected neurons with highly stochastic firing behaviors.
- The primate cortex has a hierarchy of distinct intrinsic timescales, the higher the slower.

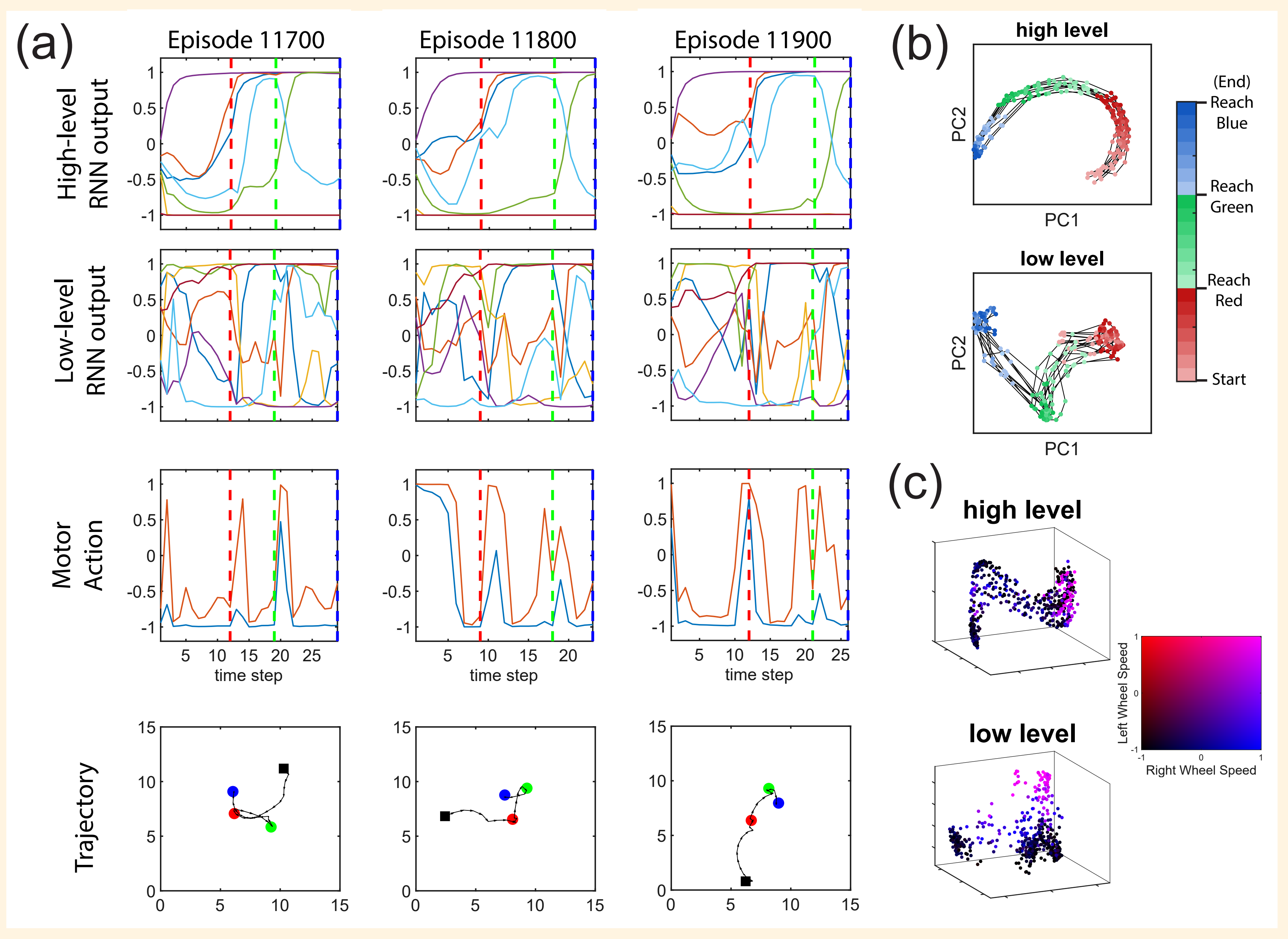
We propose ReMASTER, an RL framework in which useful actions primitives can self-organize:

- Multiple levels of RNNs with hierachical timescales, the higher the slower. (2 levels in this work)
- Stochasticity not only in the action outputs, but also in neuronal dynamics of all units.
- Only the lowest level receive observations and output actions, but each level learns a value function.

The Sequential Target-Reaching Task

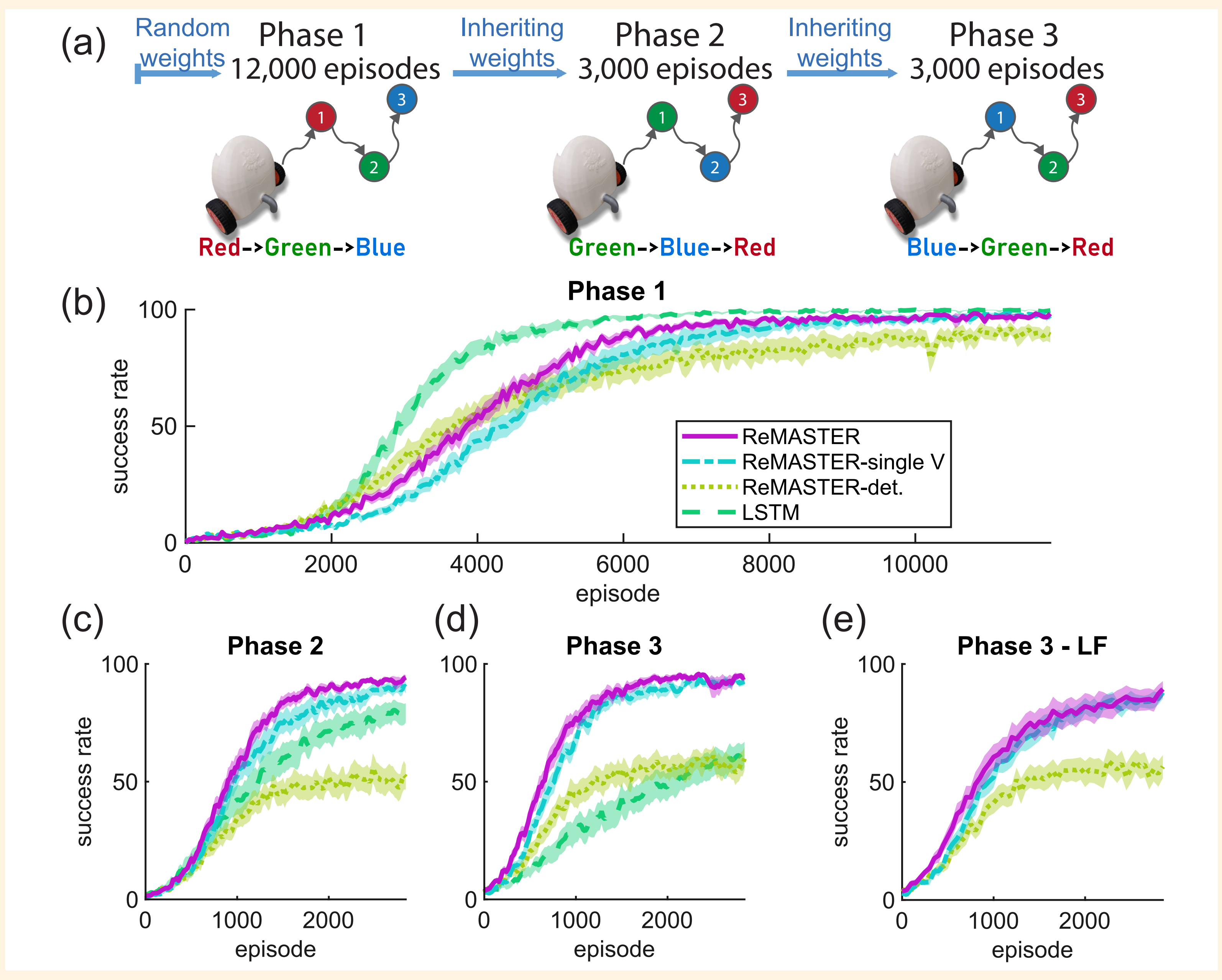


A two-wheeled robot agent (simulated) on a 2-dimensional field is required to reach three target positions, in a sequence, red-green-blue, without receiving any signal indicating which target to reach. The action is given by the rotation of its left and right wheels. At the beginning of each episode, the robot and targets are randomly placed on the field. The robot has sensors detecting distances and angles from the three targets, as well as the current-step reward. When it reaches a target in the correct sequence, it receives a one-step reward. The reward is given only if the agent followed the proper sequence, and is given only once for each target. An episode terminates if the agent completes the task or a maximum of 128 steps are taken. To successfully solve the task, the agent needs to develop the cognitive capability to remember ``which target has been reached", as well as to recognize the correct sub-goal (which can be considered as approaching a target in this task).

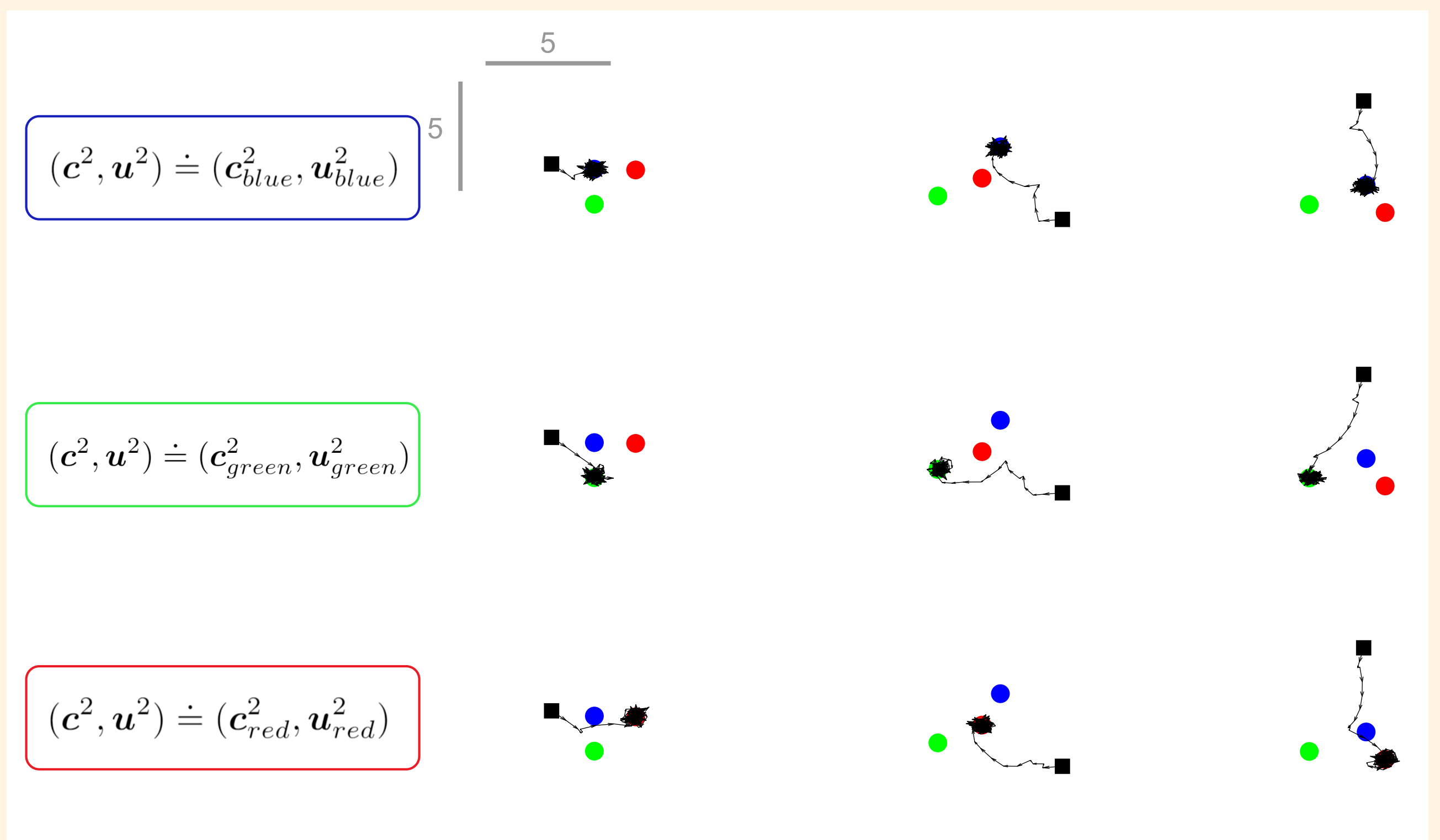


(a) Three example episodes showing the behavior of a well-trained ReMASTER agent. The first 2 rows show RNN output of two levels, where the vertical, dashed lines indicate the agent's reaching a target. The motor actions indicted by velocities of the two wheels are plotted in the third row. The fourth row is the robot's trajectories, where black squares indicate its starting positions and circles are target positions. **(b)** PCA for RNN outputs, using data from different episodes of one agent after convergence. Colors mean the agent is approaching to the corresponding targets, where a deeper color means the agent is more closed to the target. Samples from the same episode are linked with black lines. **(c)** Similar to (b), but the colors indicate speed of the two wheels (see the colormap). The first 3 PCs were plotted.

The Consecutive Relearning Task

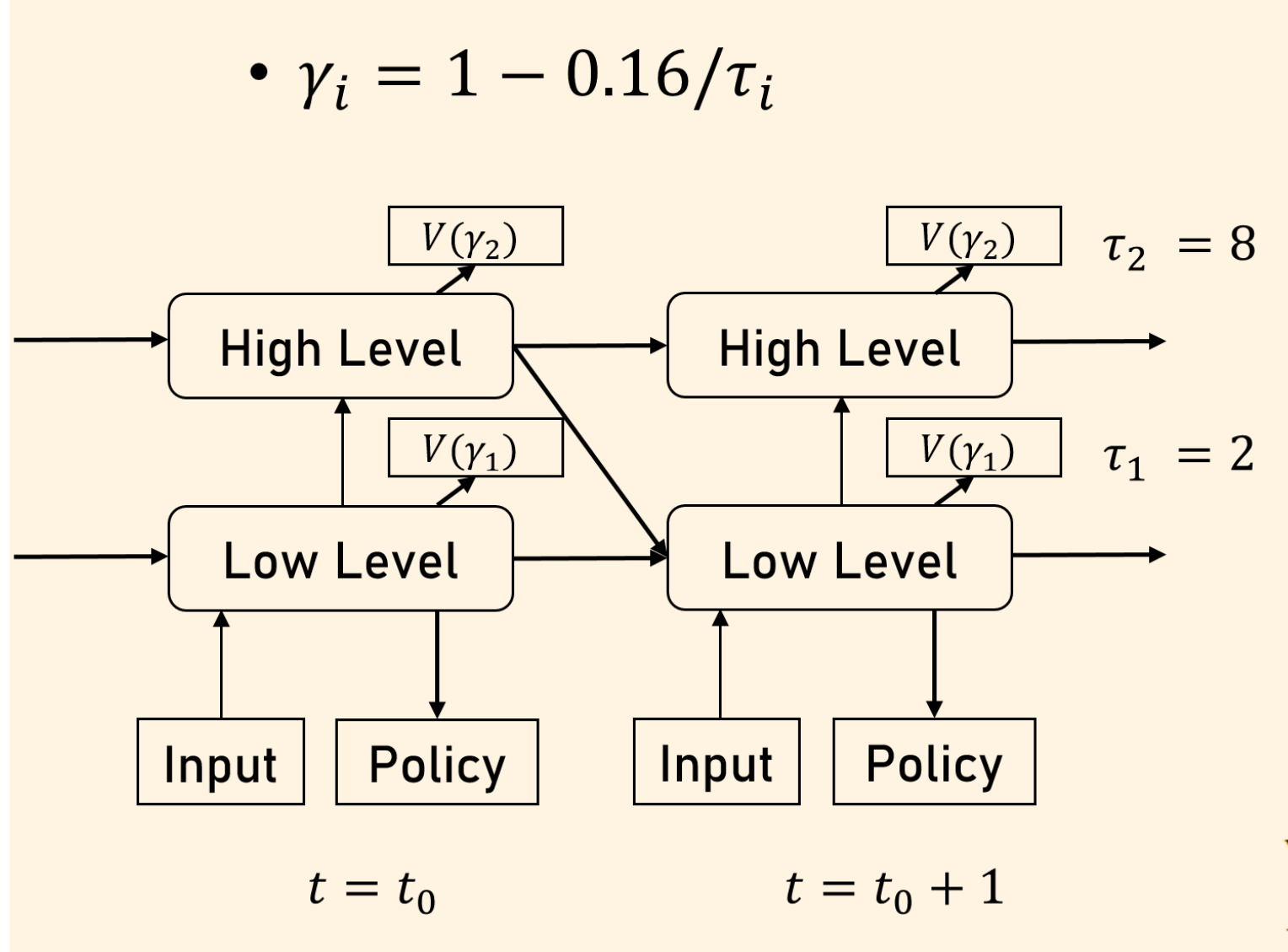


(a) Illustration of the task. **(b-d)** Performance curves for all phases. **ReMASTER-single V.** is the case in which the higher level did not learn a corresponding value function. **ReMASTER-det.** stands for the case in which all the neurons followed deterministic dynamics, and **LSTM** is the alternative using the same algorithm but the network was a single-layer LSTM. **(e)** Performance curve of phase 3 with the lower-level synaptic weights frozen (Phase 3 - LF).



Manipulating agent behaviors by clamping high-level RNN states. All trajectories were from one agent and each row used same high-level RNN states. Black squares and colored circles indicate the agent's initial positions and the target positions, respectively.

Details



Multiple Timescale Stochastic RNN (MTSRNN)

Here we describe detailed mechanisms of L-levels MTSRNN. We use a super-script $l \in \{1, 2, \ldots, L\}$ to indicate the lth level, where a smaller l indicates a lower level. Let $\boldsymbol{u}$ and $\boldsymbol{c}$ denote the *hidden states* and the *RNN outputs*, respectively[1], we have

$$\boldsymbol{u}^l(t) = (1 - \frac{1}{\tau^l})\boldsymbol{u}^l(t-1) + \frac{1}{\tau^l}\left[W_{cu}^{l-1,l}\boldsymbol{c}^{l-1}(t) + W_{cu}^{l,l}\boldsymbol{c}^l(t-1) + W_{cu}^{l+1,l}\boldsymbol{c}^{l+1}(t-1) + \boldsymbol{b}_u^l\right], \quad (1)$$

$$\boldsymbol{c}^l(t) = \tanh\left(\boldsymbol{u}^l(t) + \epsilon^l \boldsymbol{\sigma}^l(t)\right), \quad (2)$$

where $\boldsymbol{c}^{l-1}(t) = \boldsymbol{s}(t)$ when $l = 1$ is the current sensory input (state) and $\boldsymbol{c}^{l+1}$ does not exist for $l = L$. The scale of neuronal noise, σ^l, can be either a hyper-parameter or adaptive, and $\boldsymbol{\epsilon}^l(t)$ is a diagonal-covariance unit-Gaussian

Algorithm 1 ReMASTER

```
Initialize the MTSRNN $\mathcal{R}$ and the replay buffer $\mathcal{B}$, global step $t \leftarrow 0$
repeat
  Reset an episode, assign $\mathcal{R}$ with zero initial RNN states
  while episode not terminated do
    Compute 1-step forward of $\mathcal{R}$ to obtain $(\boldsymbol{u}_t^l, \boldsymbol{c}_t^l)$
    Sample an action $\boldsymbol{a}_t$ from policy $\pi_t(\boldsymbol{a}|\boldsymbol{c}_t^1)$ and execute $\boldsymbol{a}_t$
    Obtain $\boldsymbol{s}_{t+1}, r_t$ and $done_t$ from the environment
    Record $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1}, r_t, done_t)$, $\pi_t = \pi(\boldsymbol{a}_t|\boldsymbol{c}_t^1)$ and $(\boldsymbol{u}_t^l, \boldsymbol{c}_t^l)$ into $\mathcal{B}$
    if $mod(t, train\_interval) == 0$ then
      Sample sequential training samples to update $\mathcal{R}$ by
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_v \frac{1}{L}\frac{1}{N}\sum_l^L \sum_i^N \left[\rho_i \delta_i^l \nabla_{\boldsymbol{\theta}} v^l(\boldsymbol{s}_{0:t_i}; \boldsymbol{\theta})\right]$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_a \frac{1}{N}\sum_i^N \left[\rho_i \delta_i^1 \nabla_{\boldsymbol{\theta}} \log \boldsymbol{\pi}(\boldsymbol{a}_{t_i}|\boldsymbol{s}_{0:t_i}; \boldsymbol{\theta})\right]$
    end if
    $t \leftarrow t + 1$
  end while
until training stopped
```

Importance sampleing ratio

TD-error of the lth level

https://doi.org/10.1016/j.neunet.2020.06.002
GitHub.com/oist-cnru/ReMASTER