

Mental simulation on reservoir computing as an efficient planning method for mobile robot navigation

Yoshihiro Yonemura[†], and Yuichi Katori^{†‡}

[†]The School of Systems Information Science, Future University Hakodate
 116-2 Kamedanakano-cho, Hakodate, Hokkaido 041-8655, Japan

[‡]The Institute of Industrial Science, The University of Tokyo
 4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan
 Email: katori@fun.ac.jp

Abstract—Machine learning methods have been applied for autonomous mobile robot navigation. Despite the achievement of the methods, their learning cost is the most significant remaining problem. We propose a mental simulation framework on reservoir computing to perform efficient learning and action planning. Mental simulation is a process that simulates the interaction between the model and the environment. Reservoir computing is appropriate for mental simulation because it can process complex time series efficiently. In this research, we implemented action planning with mental simulation on reservoir computing, and we confirmed that the robot could reach the target point by the planning.

1. Introduction

Autonomous control is a widely researched technology that performs various tasks without external direct control. Recently, deep neural network and reinforcement learning have been applied to the autonomous mobile robot control [9]. These machine learning methods enable to construct control model with only setting a reward corresponding to the success or failure of each task. However, it requires many data. Two reasons cause this problem: deep neural networks need to optimize many parameters, and reinforcement learning requires many trials to obtain enough samples of a task.

A reservoir computing model with reinforcement learning has been proposed [3][8]. Reservoir computing can process complex time series with only learning relatively few parameters by using complex dynamics of randomly connected recurrent networks [4]. Antonelo et al. have proposed a navigation model for mobile robots based on reservoir computing and reinforcement learning [1], and the model performs navigation tasks with relatively few sensors and parameters.

We propose a mental simulation framework for reservoir computing. Mental simulation is to simulate the interaction between the model and the environment [2]. Mental simulation can perform the action planning and learning

by using the simulated experience. The reservoir computing model is appropriate for mental simulation because it can reconstruct complex novel information from its internal dynamics. Moreover, its physical implementation enables high-speed computation with high power efficiency, which is helpful for mental simulation. In the following sections, we show that the mental simulation performs action planning on the autonomous mobile robot navigation task.

2. Model and Experiments

We use the mobile robot used by Inada et al. [3] with some extensions. The robot observes distance to object, self-location, and self-direction. The robot sequentially selects one of three possible actions: turning left, turning right, and moving forward. The mobile robot is controlled by the predictive coding with reservoir computing model [5] shown in Figure 1. Predictive coding is the computational model that updates its internal state while predicting the sensory information [6].

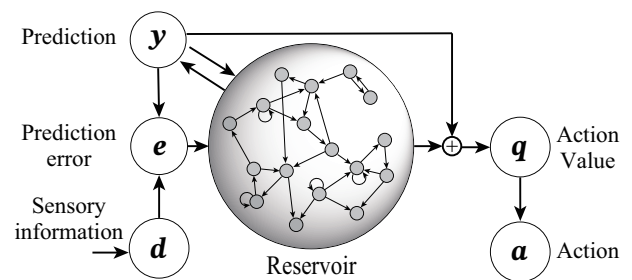




Figure 1: Reservoir computing model for autonomous control. The model is composed of six parts: dynamical reservoir, sensory information d , prediction y of the sensory information, prediction error e , action value q , and next action vector a .

The model receives sensory information $d(n)$ which is composed of three components: distance between the robot and obstacles for each direction $v \in \mathbb{R}^{32}$, self-location encoded in place cell representation $v^{(PC)} \in \mathbb{R}^{11 \times 11}$, self-direction encoded in head direction cell representation

ORCID iDs Yoshihiro Yonemura:  0000-0002-7527-233X, Yuichi Katori:  0000-0003-2773-0786

$\mathbf{v}^{(HD)} \in \mathbb{R}^{12}$, and reward $R \in \mathbb{R}$. The place cell representation $\mathbf{v}^{(PC)}$ corresponding to the self-location (x, y) is determined as following equation:

$$v_i^{(PC)} = \exp\left(-\frac{(x-x_i)^2 + (y-y_i)^2}{\sigma_{pc}^2}\right), \quad (1)$$

where (x_i, y_i) denote the center of the receptive field of i th place cell, and σ_{pc}^2 denotes width of receptive field. The head direction cell representation $\mathbf{v}^{(HD)}$ corresponding to the self-direction θ is determined as following equation:

$$v_i^{(HD)} = \exp\left(-\frac{(\theta-\theta_i)^2}{\sigma_{hd}^2}\right), \quad (2)$$

where θ_i denotes the center of the receptive field of i th head direction cell, and σ_{hd}^2 denotes width of receptive field. The model generate the prediction $\mathbf{y}(n)$ of the sensory information $\mathbf{d}(n)$ from the firing rate $\mathbf{r}(n)$ of the reservoir neurons as following equation:

$$\mathbf{y}(n) = W_{out}^{(Sensor)} \mathbf{r}(n). \quad (3)$$

The model simultaneously generates the action value $\mathbf{q}(n)$ from the firing rate $\mathbf{r}(n)$, and select next action $a(n+1)$ based on ε -greedy policy; the model selects action whose value is maximum, and selects other action with probability ε . The $\mathbf{q}(n)$ is determined by the following equation:

$$\mathbf{q}(n) = W_{out}^{(Action,1)} \mathbf{r}(n) + W_{out}^{(Action,2)} \mathbf{y}(n) + \mathbf{s}(n), \quad (4)$$

where $\mathbf{s}(n)$ denotes the context vector generated by the action planning process. The model updates its internal state $\mathbf{m}(n)$ as following equations:

$$\mathbf{I}(n) = W_r \mathbf{r}(n-1) + W_b \mathbf{y}(n-1) + \alpha_e W_e \mathbf{e}(n-1) + W_a \mathbf{a}(n), \quad (5)$$

$$\mathbf{m}(n) = \mathbf{m}(n-1) + \frac{\Delta t}{\tau} (-\mathbf{m}(n-1) + \mathbf{I}(n)), \quad (6)$$

$$\mathbf{r}(n) = \tanh(\mathbf{m}(n)), \quad (7)$$

where $\mathbf{I}(n)$ denotes the sum of input for the reservoir neurons. The Δt denotes the unit time, and τ denotes the time constant of reservoir neurons. The vector $\mathbf{a}(n) = (a_1, a_2, a_3)^T \in \mathbb{R}^3$ is the one-hot vector whose component $a_i(n)|_{i=a(n)} = 1$ and other components $a_i(n)|_{i \neq a(n)} = 0$. The connection matrices W_b , W_e and W_a are randomly initialized with connecting strength α_b , α_e , α_a , and connecting rate β_b , β_e , β_a , respectively. The spectral radius of W_r is set to α_r . The connecting rate of W_r is set to β_r . The prediction error $\mathbf{e}(n)$ is determined as following equation:

$$\mathbf{e}(n) = \mathbf{d}(n) - \mathbf{y}(n). \quad (8)$$

The readout connection $W_{out}^{(Sensor)}$ is trained by FORCE algorithm [7] as following equations:

$$P(n+1) = P(n) - \frac{P(n)\mathbf{r}(n)\mathbf{r}^T(n)P^T(n)}{1 + \mathbf{r}^T(n)P(n)\mathbf{r}(n)}, \quad (9)$$

$$W_{out}^{(Sensor)}(n+1) = W_{out}^{(Sensor)}(n) + \mathbf{e}(n)(P(n)\mathbf{r}(n))^T, \quad (10)$$

where P denotes the inverse correlation matrix of firing rate $\mathbf{r}(n)$, and it is initially set to the identity matrix. The readout connection $W_{out}^{(Action,1)}$ and $W_{out}^{(Action,2)}$ are trained by reinforcement learning which is formulated as following:

$$\delta(n) = R(n+1) + \gamma \max_a q_a(n+1) - q_{a(n)}(n), \quad (11)$$

$$W_{out,a(n)}^{(Action,1)}(n+1) = W_{out,a(n)}^{(Action,1)}(n) + \eta \delta(n) P(n) \mathbf{r}(n), \quad (12)$$

$$W_{out,a(n)}^{(Action,2)}(n+1) = W_{out,a(n)}^{(Action,2)}(n) + \eta \delta(n) \mathbf{y}(n), \quad (13)$$

where $\delta(n)$ denotes the temporal difference error of action value. The $W_{out,a(n)}^{(Action,1)}$ denotes $a(n)$ th row component of $W_{out}^{(Action,1)}$, and $W_{out,a(n)}^{(Action,2)}$ denotes $a(n)$ th row component of $W_{out}^{(Action,2)}$. The γ denotes the discount rate of the action value. The η is the learning rate. This formulation is based on the previous research [3][8], but we use $P(n)\mathbf{r}(n)$ instead of $\mathbf{r}(n)$.

The schematic of mental simulation is shown in Figure 2. The model learns on the real environment (bottom side of Figure 2), and the model searches for optimal action sequence using the simulated environment on mental simulation (upper side of Figure 2). The mental simulation process is performed by setting α_e to 0. The model does not receive the prediction error feedback and reconstructs sensory information using its internal dynamics while $\alpha_e = 0$ based on the equation (5).

The search for the optimal context is achieved by the Nelder-Mead method, which finds the minimal value of an objective function in multidimensional space. We set the five parameters as the arguments of the objective function: the initial time step of adding the context $N \in \mathbb{N}$, the period of adding context $T \in \mathbb{N}$, and the adding value $\mathbf{s} \in \mathbb{R}^3$. The context is evaluated based on the Euclidian distance between the target position of action planning and the decoded position of the predicted place cell representation. This distance is used as the objective function of this optimization. We decode the position (\hat{x}, \hat{y}) from place cell representation as following equation [10]:

$$\hat{x} = \frac{\sum_i \hat{v}_i^{(PC)} x_i}{\sum_i \hat{v}_i^{(PC)}}, \quad \hat{y} = \frac{\sum_i \hat{v}_i^{(PC)} y_i}{\sum_i \hat{v}_i^{(PC)}}, \quad (14)$$

where $\hat{v}_i^{(PC)}$ denotes the predicted place cell value. The context sequence is optimized by applying these processes 10 times. Figure 3 shows the schematic of the action value sequence with optimized context. We compare five optimized context sequences by the nearest distance in the mental simulation, and the context sequence whose score is the best is used.

In the experiments, the following parameters are used: $N_x = 500$, $\tau = 8$ (s), $\Delta t = 0.2$ (s), $\alpha_r = 0.9$, $\alpha_b = 0.1$, $\alpha_e = 0.1$, $\alpha_a = 1.0$, $\beta_r = \beta_b = \beta_e = \beta_a = 0.1$, $\eta = 0.001$, $\gamma = 0.95$, $\sigma_{pc} = 90$ (mm), $\sigma_{hd} = \pi/4$ (rad). The model is trained to predict sensory signal and avoid collision during

training. The reward corresponds to the collision; the negative reward -10 is given when the robot clashes. While the robot moves, the negative reward -0.01 is given when the robot turns, and the positive reward 0.1 is given when the robot moves forward. The parameter ε is set to 0.1 . In the testing process, the model's parameter is not configured. The parameter ε is set to 0 . The maximum time step is set to 1000 steps in the training process and 150 in the testing process.

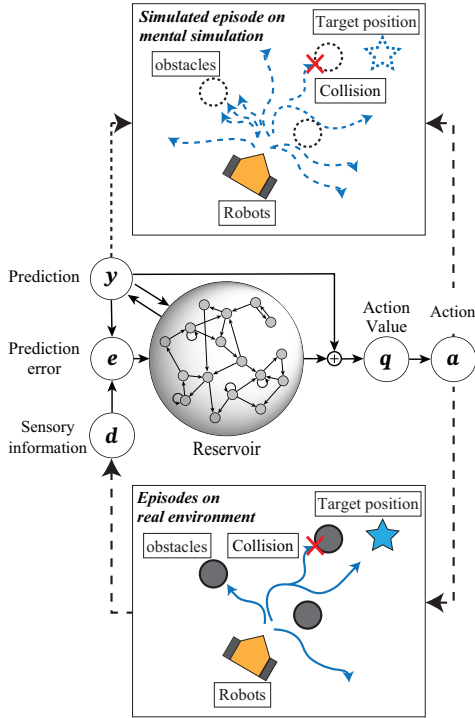


Figure 2: Schematic of mental simulation. The dashed black arrows represent the interaction between the model and the environment. The solid blue arrows represent the trajectories of the robot on real episode. The dashed blue lines represent trajectories of the robot on simulated episode.

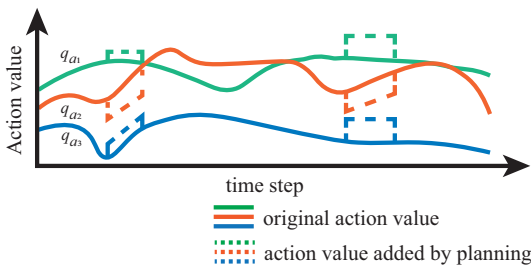


Figure 3: Schematic of action value modification. The solid lines represent the original action value. The dashed lines represent the action value added by planning process.

3. Results

Figure 4 shows the time step that the robot moves with avoiding collision for each episode. The blue dots with line represent the mean time step of 10 samples. The vertical gray line represents the standard deviation of time step. This result indicate that the robot can keep moving while avoiding collision by learning.

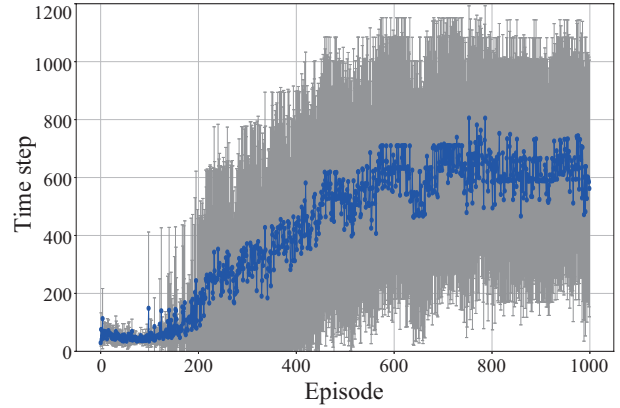


Figure 4: Time steps while the robot avoid collision. The blue line with dots represents mean time step. The gray vertical bar represents standard deviation.

Figure 5 shows the environment and the robot's trajectories. The solid black lines represent obstacles and walls. The blue lines represent the trajectory when the robot's initial direction is upper. The red lines represent the trajectory when the robot's initial direction is right. The solid blue and red lines represent the robot's trajectories without planning. The dashed lines represent the robot's trajectories with planning. The square mark represents the initial point of the robot. The star mark represents the target point of planning. Figure 5 shows that the robot successfully reaches the target point by planning (dashed lines in Figure 5). The robot does not reach the target point without planning (solid lines in Figure 5).

4. Conclusion

We proposed the mental simulation framework that performs the action planning on a mobile robot navigation task with the reservoir-based reinforcement learning model. The model performs action planning so that the robot can reach a target point without specific training for this target.

The mental simulation in the present model has two functions. The first function is to simulate the sequence of actions and the following robot's trajectories in the environment. The second function is to simulate sensory information obtained from the environment; the robot can simulate the placement of obstacles and the relationship between obstacles and the robot's position. The agent simulates how the environment around the robot will change in the next timestep from the current and previous state. The second

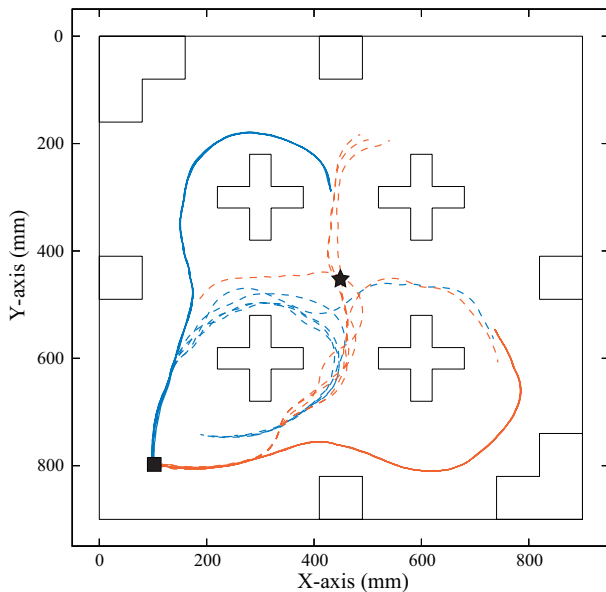


Figure 5: Trajectories of mobile robot. The blue lines represent the trajectory when the robot’s initial direction is set to upper. The red lines represent the trajectory when the robot’s initial direction is set to right. The solid lines represent the robot’s trajectories without planning. The dashed lines represent the robot’s trajectories with planning.

function requires extensive computational resources; thus reservoir computing contributes to reducing the computational cost of demanded mental simulation.

The proposed model can be applied to other reinforcement learning tasks. The simulated and optimized actions contribute to efficient performance in the tasks with the interaction between the environment and the agent. We showed that reservoir computing can process complex time-varying contextual information and is appropriate for mental simulation. Also, various physical implementation of the reservoir has been proposed. The present model should be evaluated in those various implementations in the future.

Acknowledgments

This paper is based on results obtained from a project, JPNP16007, commissioned by the New Energy and Industrial Technology Development Organization (NEDO), and this work is also supported by JSPS KAKENHI (21H05163, 20H04258, 20H00596, 21H03512), and JST CREST(JPMJCR18K2), Moonshot R&D (JPMJMS2021).

References

[1] E. A. Antonelo, D. Stefan, S. Benjamin, “Learning navigation attractors for mobile robots with reinforcement learning and reservoir computing,” *Proceedings*

of the X Brazilian Congress on Computational Intelligence (CBIC), Fortaleza, Brazil, November, 2011.

- [2] J. B. Hamrick, “Analogues of mental simulation and imagination in deep learning,” *Current Opinion in Behavioral Sciences*, vol.29, pp. 8–16, 2019.
- [3] M. Inada, Y. Tanaka, H. Tamukoh, K. Tateno, T. Morie, Y. Katori, “Prediction of sensory information and generation of motor commands for autonomous mobile robots using reservoir computing,” *Proceedings 2019 International Symposium on Non-linear Theory and its Applications (NOLTA2019)*, p.333, 2019.
- [4] H. Jaeger, “A tutorial on training recurrent neural networks, covering bppt, rtl, ekf and the “echo state network” approach,” *German National Research Center for Information Technology, ReVision*, pp.1–46, 2002.
- [5] Y. Katori, “Network Model for Dynamics of Perception with Reservoir Computing and Predictive Coding,” *Advances in Cognitive Neurodynamics (VI)*, pp.89–95, 2018.
- [6] R. P. N. Rao, D. H. Ballard, “Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects,” *Nature Neuroscience*, vol.2, no.1, pp.79–87, 1999.
- [7] D. Sussillo, L. F. Abbott, “Generating Coherent Patterns of Activity from Chaotic Neural Networks,” *Neuron*, vol.63, no.4, pp.544–557, 2009.
- [8] I. Szita, G. Viktor, L. András, “Reinforcement learning with echo state networks,” *International Conference on Artificial Neural Networks, Springer, Berlin, Heidelberg*, pp.830–839, 2006.
- [9] X. Xiao, B. Liu, G. Warnell, P. Stone, “Motion Planning and Control for Mobile Robot Navigation Using Machine Learning: a Survey,” *Autonomous Robots*, pp.1–29, 2022.
- [10] K. Zhang, I. Ginzburg, B. L. McNaughton, T. J. Sejnowski, “Interpreting neuronal population activity by reconstruction: unified framework with application to hippocampal place cells,” *Journal of neurophysiology*, vol.79, no.2, pp.1017–1044, 1998.
- [11] F. Gao, L. Han, “Implementing the Nelder-Mead simplex algorithm with adaptive parameters,” *Computational Optimization and Applications*, vol.51, pp.259–277, 2012.