Cell Tracking via Reinforcement Learning with Microscopic Image Simulator

Toru Nagamura
Graduate School of Information Science and Technology, Osaka University
Osaka, Japan

Shigeto Seno
Graduate School of Information Science and Technology, Osaka University
Osaka, Japan
senoo@ist.osaka-u.ac.jp

Hironori Shigeta
Graduate School of Information Science and Technology, Osaka University
Osaka, Japan

Tomohiro Mashita
Cybermedia Center, Osaka University
Osaka, Japan

Hideo Matsuda
Graduate School of Information Science and Technology, Osaka University
Osaka, Japan

ABSTRACT

Recent advances in optical microscopy and fluorescent protein technology have made it possible to record movies of cells over time while keeping them alive. Cell tracking is necessary to extract and analyze cell dynamics from these movies. Tracking-by-detection methods based on supervised deep learning are widely used for cell tracking. However, it is necessary to individually adjust the tracking algorithm for each cell movie that shows various characteristics and, in addition, to prepare a sufficient amount of data for training. To address these issues, we propose a method for training cell tracking models based on reinforcement learning with a simulator that imitates cell movies as an environment. The simulator can generate diverse and voluminous cell movies containing cell features from the correct trajectory of cell tracking. Through evaluation of the Cell Tracking Challenge dataset, the proposed method is confirmed to achieve a competitive performance that is better than the conventional reinforcement-learning tracker.

CCS CONCEPTS

• **Computing methodologies** → **Tracking**.

KEYWORDS

bioimage informatics, object tracking, deep reinforcement learning, biological imaging, cell movement, generative adversarial network

ACM Reference Format:
Toru Nagamura, Shigeto Seno, Hironori Shigeta, Tomohiro Mashita, and Hideo Matsuda. 2023. Cell Tracking via Reinforcement Learning with Microscopic Image Simulator. In *2023 13th International Conference on Biomedical Engineering and Technology (ICBET 2023), June 15–18, 2023, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3620679.3620682

ICBET 2023, June 15–18, 2023, Tokyo, Japan

ACM ISBN 979-8-4007-0743-8/23/06…$15.00
https://doi.org/10.1145/3620679.3620682

1 INTRODUCTION

Recent advancements in technology, including optical microscopy, fluorescent proteins, and genetic engineering, have facilitated the ability to observe living cells in real-time and capture them in movies, a field known as bioimaging. For instance, physiological processes such as immune cell-mediated wound healing[9][6]and cancer cell metastasis can now be visually observed. These movies provide valuable information on cellular dynamics, encompassing cell migration, morphology, division, fusion, adhesion, and cell death, which have significant implications for drug discovery and pathological investigation. Consequently, the extraction and analysis of cell dynamics have emerged as key technologies in medical and biological research. However, it is worth noting that bio-imaging techniques rapidly evolve towards high-throughput methodologies, resulting in the generation of various movies. As a result, there is a need to automate the extraction of cell dynamics with sophisticated image processing algorithms.

Cell tracking is the process of locating specific cells in sequential images. Typically, the input in cell tracking consists of image data, while the output includes the trajectory. The trajectory represents the precise path followed by a cell during its motion. Cell tracking is essential to quantify cell dynamics, such as cell number, morphology, division, and fusion. Because of the difficulty of performing cell tracking manually, current researches focus on developing to automate it using computers, but there are some challenges. In particular, factors such as changes in cell shape and cell division make it difficult to identify cells. In addition, the cell may appear blurred or as bright spots, making it difficult to distinguish cells in a densely populated cellular environment.

There are diverse approaches for computational cell tracking[4]. For instance, some techniques utilize probabilistic filtering methods such as particle filters and correlation filters to estimate the probability distribution of cell positions in the subsequent frame

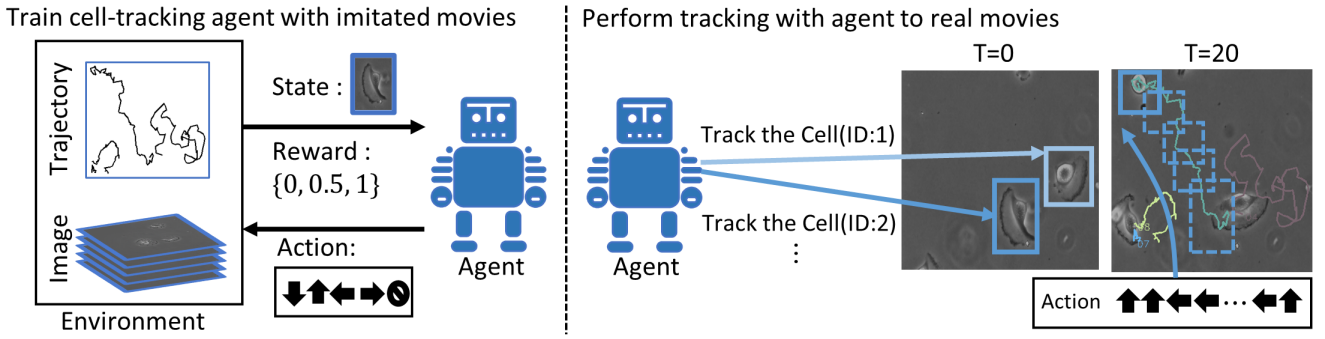

Figure 1: Reinforcement learning model for cell tracking is trained with simulator and predict trajectories on the cell movies. On the right side, the image (T=0) shows the initial location of the target. The image (T=20) shows the iterative action to track the target cell. The sequential actions control the bounding box in each frame from initial to current target cell position.

for tracking purposes[5]. Recently, a method known as tracking-by-detection, which entails detecting cells in each frame and establishing correspondences between cells that are likely to be identical, has demonstrated superior cell tracking performance. In the past, cell detection relied on Support Vector Machine (SVM) with handcrafted features[10], imposing limitations on tracking performance. However, the utilization of deep learning approaches such as Convolutional Neural Networks (CNN)[3] has significantly improved tracking performance, as it not only automates feature design but also enables detection using rich and learned feature representations[13]. Furthermore, robust tracking can be achieved across diverse data sets through transfer learning of detection models trained on large-scale data sets[11]. However, there is a hurdle to this method as the effectiveness of object tracking depends on the accuracy of the object detection results. In addition, it has another disadvantage that the appropriate tracking algorithm has to be designed for each target cell.

On the other hand, there are cell-tracking methods utilizing reinforcement learning. Reinforcement learning is an area of machine learning to get intelligent agents in an environment. The environment is the world in which the agent lives and interacts. Agent affects the environment to achieve the objective task. At its core, any reinforcement learning task is defined by the state, action, and reward. States represent the current environment. Actions are what an agent does in the current state. Rewards are feedback about actions from environment. Based on the current state, the agent takes appropriate actions, which in turn modify the state of the environment and result in the agent receiving a reward and a next state. The objective of reinforcement learning is getting optimal "policy" which calculates optimal action in a current state. The policy is obtained to maximize the expected reward value through agent-environment interaction.

To train a cell-tracking model with reinforcement learning, an environment is needed in which the correct reward can be calculated for the agent's actions. To create such an environment, conventional methods require paired data: a series of microscopic images and the trajectory of each target cell. However, generating

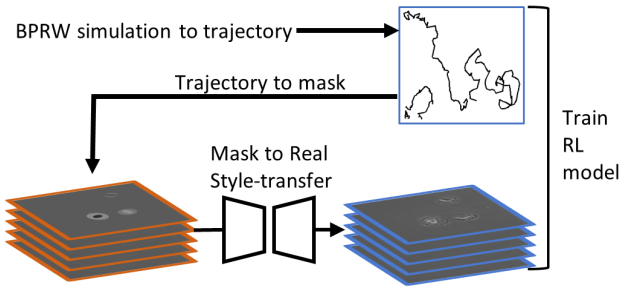

Figure 2: How to simlate cell movies

such data is an expensive task because it requires manual tracking. Preparing a sufficient number of image-trajectory pairs for reinforcement learning is impractical.

To solve this problem, we used a simulator to generate paired data. The simulator generates pseudo-microscopic images from mathematical simulation results of cell migration. Obviously, the generated dataset consists of paired data of images and cell trajectories. Thus, it is possible to compute the reward needed for reinforcement learning.

2 METHODS

In this study, we propose a method for training a reinforcement learning model for cell tracking with a simulator. The agent acquires model for cell tracking through reinforcement learning on the simulator and then, the model can be used to track target cells in an unseen video (Figure 1). The simulator generates imitated movies from mathematical simulation results of cell migration (Figure 2). Using neural style transfer architecture, the pseudo-microscopic imitated images translated from the mask images are equivalent to the actual set of images to be tracked, and also contain a variety of cellular features. The agent can learn cell tracking in the simulator as an environment, allowing it to predict trajectories in real images. The proposed approach offers a notable advantage in that the simulator can train the cell-tracking model even in the limited data availability scenario. Furthermore, the simulator automatically

adjusts the tracking algorithms to accommodate diverse cell movies, enhancing its Generalization performance.

In this section, we define cell tracking in the context of reinforcement learning, the environment and simulator, and the agent.

2.1 Setup of the Cell Tracking Problem

Reinforcement learning for cell tracking involves learning a policy that selects the optimal action for tracking the target cell based on the current state. The agent is designed to output actions that determine the current location of the target cell. During training, the agent learns to output the optimal action by taking as input the state computed from the image and the reward computed from the accurate trajectory (see Figure 1 on the left). By generating the action of locating the current position of the cell throughout the whole movie, the system finally generates the trajectory of the target cell to be tracked (see Figure 1 on the right). The objective of the learning process is to acquire a policy for generating the correct trajectory. The approach used to obtain the policy is deep reinforcement learning. We use deep reinforcement learning to obtain the optimal policy for cell tracking. In this approach, the policy is represented by a neural network and optimized by the gradient descent method.

From this point on, we will describe the problem setup of reinforcement learning for cell tracking described above. we consider a problem setting in which cells are tracked using boundig box motion. In this setting, the state, action, and reward are represented as

- State (s_t) : an image with bounding box-area croped
- Action (a_t) : indicate the moving direction of bounding box
- Reward (r_t) : evaluate action

Let I_L denote the Lth frame of the cell movie. Let s_T denote the bounding-box region extracted from the image. The policy network determines the action a_T corresponding to the bounding-box control method based on the input state s_T. The state s_{t+1} at the next time step is determined by a state transition that moves the bounding box based on the action. Furthermore, s_T and s_{T+1} are compared, and the reward r_T is calculated as a value to evaluate the chosen action. This sequence of steps is called a "step," and a total of N steps are processed in one frame. The agent takes action only N times while the tracked object moves once, gradually adjusting the position of the bounding box to capture the cell.

On this setting, each episode consists of 20 frames. an episode represents a single instance of cellular motion in the training data. Since $N = 10$ in this study, we consider 200 steps as one episode.

Our approach focuses on tracking a single cell in each episode during training, although in a real cell video tracking test, tracking is performed for every frame in which a cell is present in the image. If multiple cells are present in the image, our single cell tracking approach is repeated for each cell in order to achieve multi-cell tracking.

2.1.1 State. The state s_t corresponds to a rectangular region extracted from frame I_L of the cellular video image. The coordinates of the rectangular region are represented by a 4-dimensional vector (x_t, y_t, w, h). A coordinate system with the origin at the upper left corner of the image is utilized, where x_t and y_t denote the coordinates of the upper left corner of the bounding box at time step t, while w and h represent hyperparameters indicating the width and height of the bounding box.

2.1.2 Action. Action refers to the policy employed to manipulate the bounding box. The action space defined as five options.

- move bounding box position up or down: $\boldsymbol{a_t} = (\pm 1, 0)$
- move bounding box position left or right : $\boldsymbol{a_t} = (0, \pm 1)$
- stop bounding box : $\boldsymbol{a_t} = (0, 0)$

. the policy selects the optimal action to track the target cell.

2.1.3 State Transition. When action $\boldsymbol{a_t}$ is chosen under state s_t at time step t, the subsequent state s_{t+1} is determined based on the movement of the bounding box. The equation is that

$$(x_{t+1}, y_{t+1}) = (x_t, y_t) + \xi \boldsymbol{a_t} \tag{1}$$

where ξ is a hyperparameter dictating the amount of movement of the bounding box in pixels.

2.1.4 Reward. The reward is determined by the Euclidean distance d_t between the correct and predicted coordinates, as described in equation

$$r_t = \begin{cases} 1 & (d_t \leq \delta) \\ 0.5 & (d_t > \delta \text{ and } d_t < d_{t-1}) \\ 0 & (d_t > \delta \text{ and } d_t > d_{t-1}) \end{cases} \tag{2}$$

The hyperparameter δ signifies the tolerance for error between the correct and predicted coordinates, in pixels. Specifically, the reward is set to 1 when the bounding box is directly above the cell being tracked, 0.5 when the distance is closer than the previous step, and 0 when the distance is farther.

2.2 Environment for Cell Tracking

As described in the previous chapter, during training, the environment uses images and trajectories as input and calculates rewards for actions. The main idea of this study is to generate data in the simulator during training in this way. To facilitate reinforcement learning, we have developed a simulator that simulates the video images to be tracked and generates a substantial number of episodes. For accurate cell tracking, it is necessary to reduce the domain gap between real and simulated images. Hence, our objective is to imitate real cell migration and morphology on the simulator. In this section we describe the techniques integrated into the simulator to achieve the objective.

2.2.1 Biased Persistent Random Walk. In this study, we employ Biased Persistent Random Walk (BPRW)[8] as a computational model for simulating cell migration. BPRW is a sophisticated particle and cell migration model that incorporates the concepts of bias and persistence into the traditional Random Walk (RW) model[2]. RW is a widely used model for describing cell migration, where the movement of one particle is influenced by another particle, referred to as "bias," and the movement of a particle is persistently biased towards maintaining its previous direction of migration, referred to as "persistence." In comparison to RW, BPRW offers a more accurate and refined representation of cell migration.

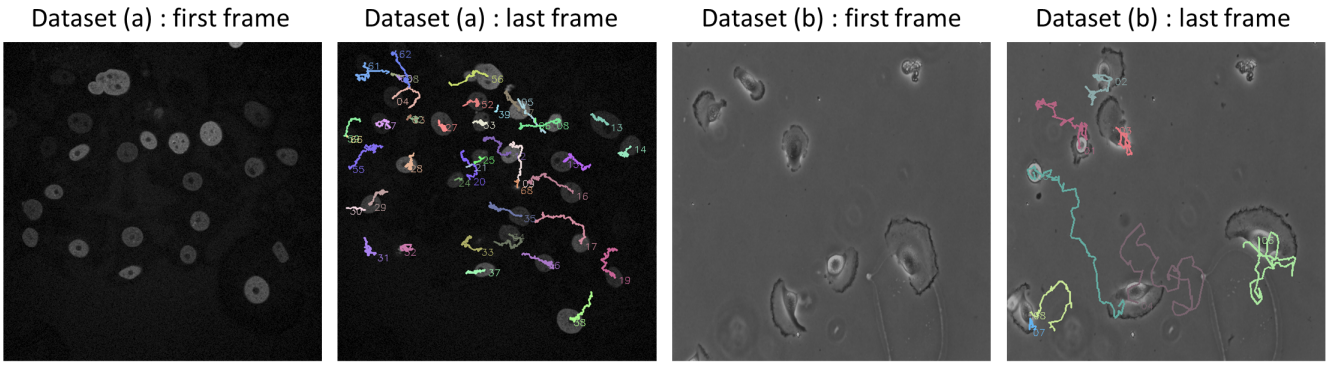

Figure 3: Example of experimental dataset. The first and last frames of each dataset are depicted in this figure. Each cell trajectory is drawn by colored line on the last frame.

2.2.2 simulator. We employ CycleGAN[17] to generate images within the simulator that imitate migration and morphology patterns with real images. First, we create a "mask image" with ellipses and other cell-like shapes positioned randomly. Second, we train a CycleGAN model capable of transforming the "mask image" into a realistic image. Finally, we generate a "mask image" as a movie that captures a specific cell migration trajectory, and transform it using CycleGAN. Through this approach, the simulator is able to generate movies that imitate temporal cell migration and morphological changes.

2.2.3 How to train the CycleGAN. To train CycleGAN to replicate cell morphology, "mask images" are generated, as shown in Figure4 top row, for datasets (a) and (b), respectively. The images that are not used in the tracking experiment are the training dataset of CycleGAN. Dataset (a) contains 182 images and dataset (b) contains 228 images. They are augmented 6 times larger by rotating them by 90, 180, and 270 degrees and by flipping them vertically and horizontally. Cell migration is assumed to follow BPRW for both datasets. An example image generated by the proposed procedure is shown in Figure4 bottom row.

2.2.4 Image Processing for Reinforcement Learning. In order to optimize the efficiency of reinforcement learning, two additional elements are incorporated into the generated cellular moving images. Firstly, a checkerboard pattern is introduced into the background of each frame. As compared to a uniform background color, this allows the agent to discern and establish the relationship between the action and the rectilinear motion by comparing the checkerboard pattern of the previous and current frames. Additionally, a cross is drawn to denote the center of the bounding box. This facilitates the agent's comprehension that a reward is obtained when the cross and the cell overlap.

2.3 Agent for Cell Tracking

2.3.1 Deep Reinforcement Learning and Policy Learning. The agent endeavors to acquire an optimal policy for cell tracking from the state-reward pairs. In this investigation, we employ Proximal Policy Optimization (PPO)[15], a form of policy gradient method[14], as the algorithm for policy acquisition through reinforcement learning. The policy gradient method is a type of deep reinforcement learning that combines the principles of reinforcement learning and deep learning, and optimizes the policy represented by a neural network using the gradient descent method.

The policy π_θ is modeled by a policy network with parameter θ, which we refer to as the policy network. Let

$$\tau = (s_0, r_0, a_0, s_1, r_1, a_1, \cdots, s_T, r_T, a_T) \tag{3}$$

denote a time series of state, action, and reward experience data obtained when an agent has policy π_θ. The discounted reward sum, referred to as earnings, is defined as follows, where γ is the discount rate.

$$G(\tau) = r_0 + \gamma r_1 + \gamma^2 r_2 \cdots + \gamma^T r_T. \tag{4}$$

The objective of the policy gradient method is to optimize the parameter θ in order to obtain a policy that maximizes the returns. Since the returns are stochastic, the objective function of the neural network is

$$J(\theta) = E_{\tau\sim\pi_\theta}[G(\tau)], \tag{5}$$

where the measure π_θ is a probability distribution that selects an action based on the current state. Additionally, τ follows a probability distribution π_θ and E represents the expectation operator. The gradient is

$$\begin{aligned} \nabla J_\theta(\theta) &= \nabla_\theta E_{\tau\sim\pi_\theta}[G(\tau)] \\ &= E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} G(\tau)\nabla \log \pi_\theta(a_t, s_t)\right]. \end{aligned} \tag{6}$$

To update the parameters of the neural network, we can utilize the gradient $\nabla J_\theta(\theta)$, which is

$$\theta = \theta + \omega \nabla J_\theta(\theta), \tag{7}$$

where ω represents the learning rate.

There are two approaches for updating neural networks using the policy gradient method: experience replay and frame stacking[12]. In experience replay, mini-batches are randomly sampled from the empirical data pre-stored in a buffer during training. This mitigates data bias between different mini-batches, resulting in stable learning. On the other hand, frame stacking involves overlapping multiple consecutive frames and feeding them as a single state

Table 1: Hyperparameters of agent in an experiment

Parameter	Values
Total number of learning steps	100,000
Experience replay buffer size	4,096
Mini batch size	512
Number of epochs	8
Reward discount rate (γ)	0.99
Learning rate(ω)	0.00025
bounding box move(ξ)	2 (pixel)
limit of reward prediction error(δ)	6 (pixel)

input to the neural network. This approach is essential when stacking multiple states, such as in video frames, in order to select the optimal action.

2.3.2 Policy network architecture. The policy network comprises of three convolutional layers and one fully connected layer. Softmax functions are applied in the output layer, while ReLU (Rectified Linear Unit) functions are utilized in the other layers. The parameters of each layer are initialized randomly. The input image is a stack of three consecutive frames acquired from the video output of the simulator. As each frame is an RGB image, the total number of channels is 9. Image normalization is employed as a preprocessing step. The output layer generates a probability distribution for the action. The remaining hyperparameters necessary for training are detailed in Table1. The mini-batch size in the table indicates the amount of data used for a single gradient update, and the number of epochs signifies the frequency at which mini-batches are employed for gradient updates.

3 RESULTS AND DISSCUSSION

3.1 Experimental setup

Experiments about cell tracking on two datasets are conducted to confirm the efficacy of the proposed approach. The comparative methods are itemized below.

- ADNet[16] : An object tracking method based on reinforcement learning.
- Baseline : Proposed method traied by only real image.
- Ours (without CycleGAN) : Proposed method but CycleGAN is not used in the simlator.
- Ours (with CycleGAN) : Proposed method that CycleGAN is used in the simlator.

In ADNet, a pre-trained model for general object tracking is transferred to cell tracking, and the training is solely carried out on real images. Conversely, the proposed method without CycleGAN constructs a simulator by overlaying single cell images extracted from the ground truth onto the ellipse on a "mask image".

To assess the effectiveness of the simulator, training is performed on a dataset consisting of simulator-generated images mixed with real images, with varying percentages of simulator-generated images, namely 25%, 50%, 75%, and 90%. This percentage denotes the total number of simulator-generated videos relative to the total number of videos utilized for training. For instance, if there are 10 real images and the simulator percentage is set at 50%, there will be 10 simulator-generated images. By extracting the 20 frames required for an episode from the video images present in each dataset, approximately 360 episodes can be generated for dataset (a), and 77 episodes for dataset (b). Based on this, the number of simulator-generated images is determined proportionally.

Table 2: Overview of datasets

Dateset	(a)	(b)
Image size (pixel)	1,024×1,024	696×520
Pixel size (μm)	0.24	0.65
Number of videos	2	2
Interval between shots (minutes)	5	15
Number of frames	91	114
Number of cells with GT	58, 28	8, 6

3.1.1 Dataset. Illustrative examples of images from the two datasets utilized in our experiment are presented in Figure3. The dataset (a) and (b) consist of time-lapse images of stem cells from GFP-GOTW1 mice and stellate tumor cells cultured on polyacrylamide substrate, respectively. Detailed information about the dataset is provided in Table2. Both datasets are in grayscale and were resized to 512 × 512 pixels, preprocessed with gamma correction, and used as the green channel of the RGB image for training and prediction purposes. Every dataset possesses a correct trajectory. During training with real images, we use this information. The experiment was conducted through the process of cross-validation.

3.1.2 Evaluation Metrics. We adopt the One Pass Evaluation (OPE) metric for assessing the performance of single-object tracking. OPE measures the percentage of matched cells in a given tracking sequence, where a match is defined as the Euclidean distance between the center coordinates of the true and predicted values in a frame being less than a certain threshold value. In this study, the threshold was set to 20 pixels for dataset (a) and 30 pixels for dataset (b).

3.1.3 Implementation. All simulators and agents involved in the proposed method are implemented using the Python programming language and the reinforcement learning framework OpenAI Gym and stable baselines3 [1]. All experiments are conducted on a computer equipped with an Intel Core i9-10900X CPU @ 3.70GHz, 64GB RAM, and a Geforce RTX2080 GPU.

3.2 Results

Figure 5 shows the experimental results. In both datasets, ours outperforms the ADNet and baseline OPEs when the percentage of simulator images is 50% and 75%, respectively. In dataset (a), ours (with CycleGAN) recorded the third highest OPE, about 83%. dataset (b), ours(with CycleGAN) achieved the largest OPE among the comparison methods, approximately 81%.

The fact that ours outperforms ADNet and baseline confirms the effectiveness of the proposed method for cell tracking.

3.3 Discussion

Our proposed method with CycleGAN, has demonstrated superior performance compared to ADNet and the baseline, substantiating

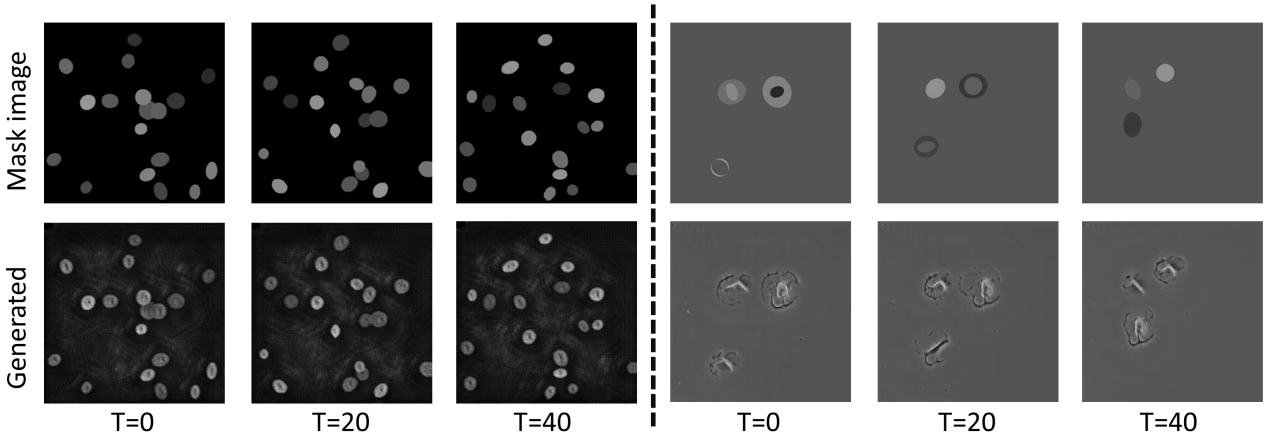

Figure 4: CycleGAN-generated images. Dataset (a) is on the left and dataset (b) is on the right. The mask image in dataset (b) has two overlapping circles indicate the cell membrane and nucleus.

its effectiveness in our experimental evaluation. Especially, the proposed method was more successful for dataset (b). Cells in dataset (a) have less cell migration and deformation. In addition, cells and background are also easy to distinguish. This means that cell tracking in dataset (a) is relatively easy. Therefore, the effect of increasing the variation of the training data via the simulator was considered to be limited. On the contrary, the cells in dataset (b) exibit a more complex structure, such as nuclei and membranes. Cells move more drastically and form differently. The contrast between cells and background is not sufficient. These results suggest that our method is more effective in situations where tracking is difficult. The simulator leads to better training in the situation because it can increase variation of the data.

We also found that the proportion of simulated images in the training data also affects the tracking accuracy. Higher accuracy can be achieved by mixing real data and imitated data in a reasonable proportion. Including imitated data increases the number of episodes that can be used for training. This leads to a gradual improvement in accuracy because a wider variety of patterns can be learned. On the other hand, too much ratio of imitated data results in decreased accuracy. The reason is that the imitated images generated by the simulator deviate from the real data more or less. The accuracy was improved by using the data generated with CycleGAN rather than the data generated without CycleGAN (just copying and pasting texture of cells). These results indicatesd that there is a certain significance in extending the information on the appearance of cells, and there is a possibility of further improvement in accuracy by using a more powerful simulator.

4 CONCLUSION

In this study, we have proposed a novel approach for training cell tracking models using reinforcement learning via a simulator that simulates cell movies. We have introduced a method using CycleGAN, which transforms the "mask image" into a realistic image to create the simulator. The simulator is capable of generating an infinite number of videos containing distinctive cell features and accurate trajectories. This simulator acts as an environment to calculate the reward for the action chosen by the agent. This approach provides a robust framework for reinforcement learning, allowing us to depart from the conventional supervised learning paradigm that focuses on determining cell identity across successive frames. Instead, our method approaches the problem from the perspective of acquiring an intelligent agent that can select appropriate actions for tracking a target. Remarkably, the agents acquired by our proposed method demonstrate performance comparable to existing methods developed for object tracking.

Future objectives are the integration of additional action types and refinement of the simulator construction method. In the proposed approach, tracking is executed through an agent that selects diverse actions. The principal advantage of this technique is the potential to adapt to varying circumstances by adding a repertoire of actions. Cells have diverse dynamics beyond mere migration, including division, fusion, cell death, and adhesion. To extract these cynamics with supervised learning, significant modifications, such as algorithmic design and network incorporation for dynamics detection, are needed. However, our proposed approach, which employs a simulator and reinforcement learning, has an ability to detect cell dynamics without requiring fundamental alterations to the method. The reason is that defining the states, actions, and rewards leads to obtaining the optimal policy to extract the multiple and complex cell dynamics.

Furthermore, there is considerable room for enhancement in the construction of the simulator. In the experiments detailed in this manuscript, migration was characterized solely by a single model denoted as BPRW. Although this model is efficacious for microscopic observation of cultured cells, it falls short in comprehensively capturing cell dynamics in complex environments such as in vivo. Consequently, we intend to estimate a cell migration model through employment of a simulator, wherein the utilization of SimGAN [7] holds potential. This approach not only broadens

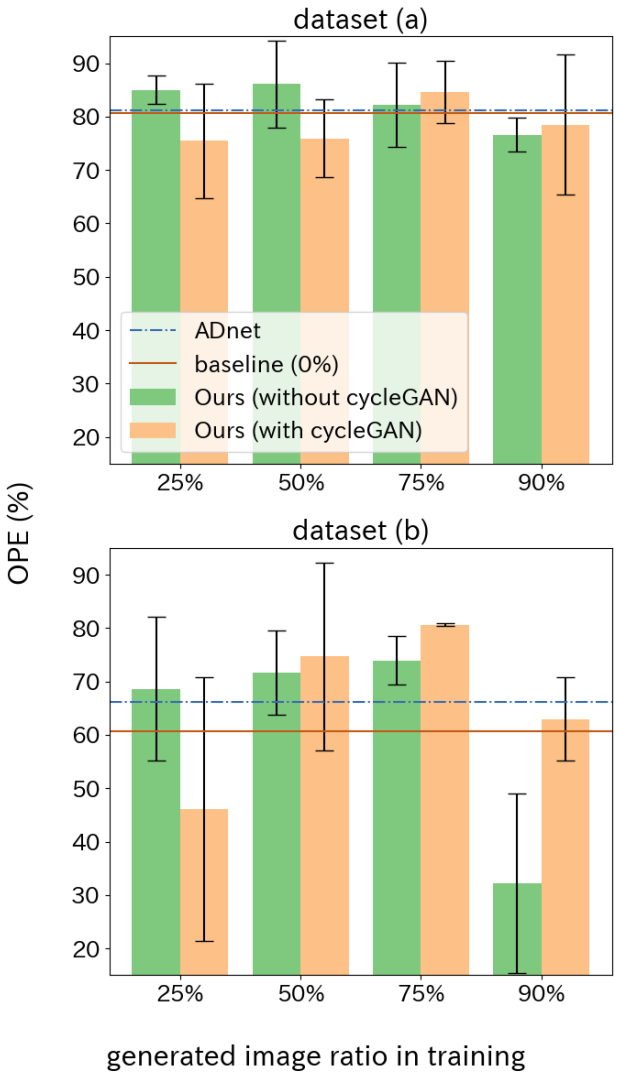

Figure 5: Expermental results

the applicability of the methodology, but also enhances tracking performance as the simulator generates realistic images, thereby minimizing the domain gap between the simulator and reality.

ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI Grant Number JP18K19842, JP22H05085.

REFERENCES

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba Openai. 2016. OpenAI Gym. (jun 2016). https://doi.org/10.48550/arxiv.1606.01540 arXiv:1606.01540

[2] Edward A. Codling, Michael J. Plank, and Simon Benhamou. 2008. Random walk models in biology. *Journal of the Royal Society Interface* 5, 25 (2008), 813–834. https://doi.org/10.1098/rsif.2008.0014

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[4] Neda Emami, Zahra Sedaei, and Reza Ferdousi. 2021. Computerized cell tracking: Current methods, tools and challenges. *Visual Informatics* 5, 1 (2021), 1–13. https://doi.org/10.1016/j.visinf.2020.11.003

[5] Kenji Fujimoto, Shigeto Seno, Hironori Shigeta, Tomohiro Mashita, Masaru Ishii, and Hideo Matsuda. 2020. Tracking and analysis of fucci-labeled cells based on particle filters and time-to-event analysis. *IJBBB* (2020).

[6] Yusri Dwi Heryanto, Chin-Yi Cheng, Yutaka Uchida, Kazushi Mimura, Masaru Ishii, and Ryo Yamada. 2021. Integrated analysis of cell shape and movement in moving frame. *Biology Open* 10, 3 (mar 2021). https://doi.org/10.1242/bio.058512

[7] Yifeng Jiang, Tingnan Zhang, Daniel Ho, Yunfei Bai, C. Karen Liu, Sergey Levine, and Jie Tan. 2021. SimGAN: Hybrid Simulator Identification for Domain Adaptation via Adversarial Reinforcement Learning. (oct 2021), 2884–2890. https://doi.org/10.1109/ICRA48506.2021.9561731 arXiv:2101.06005

[8] Phoebe J.M. Jones, Aaron Sim, Harriet B. Taylor, Laurence Bugeon, Magaret J. Dallman, Bernard Pereira, Michael P.H. Stumpf, and Juliane Liepe. 2015. Inference of random walk models to describe leukocyte migration. *Physical biology* 12, 6 (sep 2015). https://doi.org/10.1088/1478-3975/12/6/066001

[9] Daniel Kreisel, Ruben G. Nava, Wenjun Li, Bernd H. Zinselmeyer, Baomei Wang, Jiaming Lai, Robert Pless, Andrew E. Gelman, Alexander S. Krupnick, and Mark J. Miller. 2010. In vivo two-photon imaging reveals monocyte-dependent neutrophil extravasation during pulmonary inflammation. *Proceedings of the National Academy of Sciences of the United States of America* 107, 42 (2010), 18073–18078. https://doi.org/10.1073/pnas.1008737107

[10] Shohei Kumagai and Kazuhiro Hotta. 2015. Particle detection in intracellular images and radius estimation by circle fitting. *IEEJ Transactions on Electrical and Electronic Engineering* 10, 2 (2015), 181–185.

[11] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. 2022. Trackformer: Multi-object tracking with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 8844–8854.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. (2013), 1–9. arXiv:1312.5602 http://arxiv.org/abs/1312.5602

[13] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

[14] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. 2015. Trust Region Policy Optimization. *32nd International Conference on Machine Learning, ICML 2015* 3 (feb 2015), 1889–1897. https://doi.org/10.48550/arxiv.1502.05477 arXiv:1502.05477

[15] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov Openai. 2017. Proximal Policy Optimization Algorithms. (jul 2017). arXiv:1707.06347 https://arxiv.org/abs/1707.06347v2

[16] Sangdoo Yun, Jongwon Choi, Youngjoon Yoo, Kimin Yun, and Jin Young Choi. 2017. Action-decision networks for visual tracking with deep reinforcement learning. *Proceedings – 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-Janua (2017), 1349–1358. https://doi.org/10.1109/CVPR.2017.148

[17] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.