CAAI Transactions on Intelligence Technology

WILEY

The Institution of Engineering and Technology

ORIGINAL RESEARCH OPEN ACCESS

Robotic Cell Micromanipulation Skill Learning via Imitation-Enhanced Reinforcement Learning

Youchao Zhang[1] | Fanghao Wang[1] | Xiangyu Guo[1] | Yibin Ying[1] | Mingchuan Zhou[1] | Zhongliang Jiang[2] | Alois Knoll[3]

[1]Key Team of Intelligent Bioindustry Innovation Team, Department of Biosystems Engineering, Zhejiang University, Hangzhou, China | [2]Department of Mechanical Engineering, University of Hong Kong, Hong Kong, Hong Kong | [3]School of Computation, Information, and Technology, Technical University of Munich, Munich, Germany

Correspondence: Mingchuan Zhou (mczhou@zju.edu.cn)

Received: 11 October 2024 | **Revised:** 20 May 2025 | **Accepted:** 22 July 2025

Keywords: deep learning | intelligent robots | intelligent systems | robotics

ABSTRACT

Humans can learn complex and dexterous manipulation tasks by observing videos, imitating and exploring. Multiple end-effectors manipulation of free micron-sized deformable cells is one of the challenging tasks in robotic micromanipulation. We propose an imitation-enhanced reinforcement learning method inspired by the human learning process that enables robots to learn cell micromanipulation skills from videos. Firstly, for the microscopic robot micromanipulation videos, a multi-task observation (MTO) network is designed to identify the two end-effectors and the manipulated objects to obtain the spatio-temporal trajectories. The spatiotemporal constraints of the robot's actions are obtained by the task-parameterised hidden Markov model (THMM). To simultaneously address the safety and dexterity of robot micromanipulation, an imitation learning optimisation-based soft actor-critic (ILOSAC) algorithm is proposed in which the robot can perform skill learning by demonstration and exploration. The proposed method is capable of performing complex cell manipulation tasks in a realistic physical environment. Experiments indicated that compared with current methods and manual remote manipulation, the proposed framework achieved a shorter operation time and less deformation of cells, which is expected to facilitate the development of robot skill learning.

1 | Introduction

Robotic micromanipulation technology is one of the key technologies in micron-level automation [1]. Applications of micromanipulation are numerous such as microassembly of mechanical components [2], handling of biological samples [3] or microsurgery [4]. Biological cells are a class of fragile and deformable objects. Multiple end-effectors manipulation of free micron-sized deformable cells is one of the challenging tasks in robotic micromanipulation because of the complexity of their contact dynamics [5].

The contact between the end-effectors and the cell is usually modelled as an underactuated system with an uncertain response [6]. As the viscoelastic soft body, cells are prone to deformation when pressurised by a rigid body, but excessive deformation can cause irrecoverable damage to the cells, which is unacceptable [4]. In robotic cell manipulation (e.g., transport [7], characterisation [8] and surgery [9]), cell deformation must be minimised to avoid cell damage. Therefore, manual control of the remote joystick for cell manipulation requires specialised skills. Moreover, the process can be tedious and time-intensive. Achieving the desired outcome might take numerous attempts,

and it is challenging to guarantee accuracy due to the intricate nature of cellular structures and the sensitivity of the manipulation process.

Robotic cell micromanipulation (cell posture control and cell injection) involves the cooperative work of multiple end-effectors. Cell orientation control and injection are key procedures for cell surgery tasks such as intracytoplasmic sperm injection, embryo biopsy, prokaryotic cell transfer and cell transfection. Before cells are injected, organelles that affect developmental viability need to be moved away from the injection path. For example, in mammalian oocyte manipulation, the polar body of the oocyte must be rotated away from the injection site to prevent damage to the spindle within the oocyte near the polar body [4]. When performing suspension cell injection manipulation, it needs to capture the cell using a holding micropipette, and when the injection needle stabs the cell, both end-effectors need to contact the cell at the same time. The synergistic manipulation of multiple end-effectors requires the dexterity of the robot [3].

Current cell micromanipulation work predominantly employs traditional control methodologies. Dai et al. established an approximate mechanical model and an underdriven motion model of the cell and realised the attitude control of deformable oocytes by an optimal control algorithm [4]. This method relies on the mechanical model but there are parameter differences between different cells. Zhuang et al. proposed a three-loop rotary and positioning optimal control framework that can precisely realise the detection and positioning of the injection needle [3]. The proposed method is not applicable to the cooperative control of multiple end-effectors. There have been attempts to use fluid flow caused by pressure or vibration to rotate cells. However, the orientation control accuracy is easily affected by the flow inertia and the position of the cell in the fluid flow field. Optical tweezers use a focused laser beam to apply a rotational force to the cell, but may cause photodamage to the cell [10]. Orientation control using dielectrophoretic forces has also been demonstrated, but the effect of the applied electric field on cell development is unknown [11]. The drawbacks of traditional micromanipulation methods encompass the requirement for modelling variable-parameters interaction process, difficulty in flexibly controlling multiple end-effectors and the risk of inflicting damage on cells. In contrast, the proposed imitation learning method is a dexterous manipulation strategy independent of the physical model, which can ensure the safety of deformable fragile cells by flexibly controlling the end-effector.

In imitation learning, demonstrations of successful behaviours are used to train strategies to imitate the experts who provide these successful trajectories [12, 13]. Behavioural cloning is a simple approach to imitation learning that learns demonstrated behaviours through supervised learning. Although behavioural cloning has been successfully applied in a variety of domains, purely imitation learning approaches fail to exceed the demonstrator's capabilities because of the lack of the concept of task performance [14]. Imitation learning can be utilised to bootstrap the reinforcement learning (RL) process [15]. Bootstrapping helps to overcome exploration challenges, whereas RL fine-tuning allows for improved strategies based on actual task goals. In recent imitation learning work, Hesterd et al. utilised demonstrations to pre-train Q-functions by minimising the TD error [16]. In addition to this, demonstrations have been used to guide exploration, but these demonstrations are usually rule-based or in discrete spaces, making them difficult to apply in real scenarios [17]. To make effective use of the demonstration information, Vecerik et al. proposed the DDPGfD method, which optimises the experience replay method of DDPG, and the demonstration information is incorporated to the replay buffer [18]. This method can enhance sample efficiency to a certain extent, but limits the exploration space of the strategy. Rajeswaran proposed the DAPG method, which improves the gradient updating strategy of RL, and balances the relationship between demonstration and exploration using a heuristic weighting scheme [19]. In addition to state-action imitation, Radosavovic et al. proposed the state-only imitation learning algorithm (SOIL) [20]. SOIL extends DAPG to a state-only imitation setting, where an inverse model is learnt using the collected trajectories while running the strategies. This inverse model predicts missing movements in the presentation and trains the strategy like DAPG. Both the strategy and the inverse model are optimised. Hafez et al. proposed a self-supervised task inference approach, which learns action and intention embeddings from self-organisation of the observed movement and effect parts of unlabelled demonstrations and a higher level behaviour embedding from self-organisation of the joint action–intention embeddings [21]. Wan et al. proposed LOTUS, which uses an open-vocabulary vision model and extracts skills as recurring patterns presented in unsegmented demonstrations [22]. However, aforementioned imitation learning approaches have difficulty in guaranteeing the safety of the operational task because these state-of-the-art methods lack the spatiotemporal constraints extracted from the demonstration.

Furthermore, skill learning for robotic cell micromanipulation currently has a less effective framework that has been proposed. Mori et al. used Gaussian mixture model (GMM) to obtain the ideal pipette trajectory and used long short-term memory (LSTM) to infer the pipetting operation at the next time step to assist the expert operation [23]. However, this approach still requires an expert to operate remotely. Unlike general macro-robots, since robotic micromanipulation is performed at the microscopic scale, it is hard to manually drag a robotic arm to accomplish the acquisition of a large number of effective trajectories [8]. Robotic micromanipulation usually relies on expert-controlled teleoperators for master-slave control, a time-consuming and inefficient process with low safety and a tendency to damage the manipulated micro-objects. The trajectories of multiple end-effectors at the micrometre scale are hard to directly be acquired, so trajectory extraction and collection pose another challenge for robot cell micromanipulation skill imitation learning.

Humans can learn manipulation tasks through observation, imitation and exploration [24]. Human learning process facilitates the acquisition of sophisticated and dexterous manipulation skills while minimising the risk of damage to the object, making it ideal for dexterous cell micromanipulation tasks. Inspired by the human learning process, we propose an imitation-enhanced reinforcement learning method inspired by the human learning process that enables robots to learn cell

micromanipulation skills from videos. For observation, the MTO network is designed to identify the two end-effectors and the manipulated objects to obtain the spatiotemporal trajectories. For imitation, the spatiotemporal constraint in expert demonstrations is extracted by THMM to ensure the cell micromanipulation skill safety. For exploration, the ILOSAC algorithm was proposed in which the robot can perform dexterous skill learning by demonstration and exploration. The key contributions are summarised as follows:

1. A pipeline for robot cell micromanipulation imitation learning through demonstration videos is proposed. Spatiotemporal trajectories of multiple end-effectors as well as manipulated micro-objects are extracted from expert videos by designing a multi-task state estimation network. The expert videos were converted into micromanipulation demonstrations through state and action extraction.
2. An imitation learning optimisation-based soft actor-critic method for robotic cell micromanipulation skill learning is proposed. The spatiotemporal constraint in expert demonstrations is extracted by the task-parameterised hidden Markov model. And the policy performs dynamic skill learning by combining demonstration and exploration.
3. Through the proposed pipeline, the policy can complete the corresponding cell micromanipulation tasks in a real environment. Experiments indicated that compared with manual remote manipulation, the proposed framework achieved a shorter operation time and less deformation of cell, which is expected to facilitate the development of robot skill learning.

2 | Methods

As depicted in Figure 1, we propose an imitation learning framework inspired by the human learning process that enables robots to learn dexterous micromanipulation skills from videos. Firstly, for the microscopic robot micromanipulation videos, the MTO is designed to identify the two end-effectors and the manipulated objects to obtain the spatiotemporal trajectories. The spatiotemporal constraints of the robot's actions are obtained by the THMM. To simultaneously address the safety and dexterity of robot micromanipulation, the ILOSAC algorithm was proposed in which the robot can perform skill learning by demonstration and exploration. The proposed method is capable of performing complex cell manipulation tasks in a realistic physical environment.

2.1 | Multi-Task Observation From Videos

In order to extract the demonstration information from the expert video of the robot cell micromanipulation, we utilise a multitasking state detection network, as shown in Figure 2. The MTO is able to detect both end-effectors of the micromanipulation robot as well as the state information of the manipulated object (free-moving, deformable) at the same moment, as depicted in Figure 3. The positional coordinates of the tips of the two end-effectors e_1, e_2, as well as the position information e_3, orientation ω, and mask m_c of the manipulated object, can be obtained from the output of the MTO network. The size and shape of the mask partially reveal the phenotypic and state information of the cells. The changes in the relative position

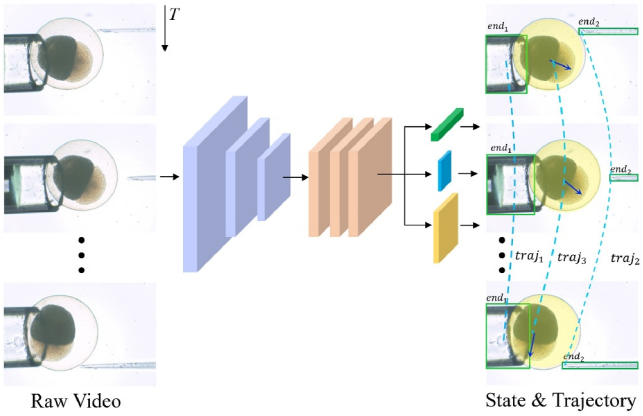


FIGURE 2 | The multitasking observation network (MTO) is able to detect both end-effectors of the micromanipulation robot as well as the state information of the manipulated object (free-moving, deformable) at the same moment. The positional coordinates of the tips of the two end-effectors e_1, e_2, as well as the position information e_3 and mask m_c of the manipulated object, can be obtained from the output of the MTO network.

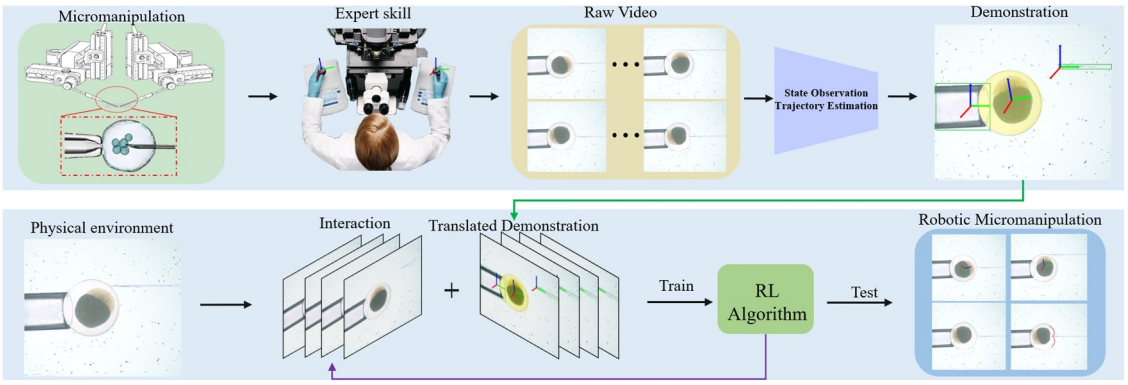


FIGURE 1 | Overall architecture. A pipeline for robot micromanipulation imitation learning through micromanipulation videos is proposed. Spatiotemporal trajectories of multiple end-effectors as well as manipulated micro-objects are extracted from expert videos by designing a multi-task state estimation network. The expert videos were converted into micromanipulation demonstrations through state and action extraction.

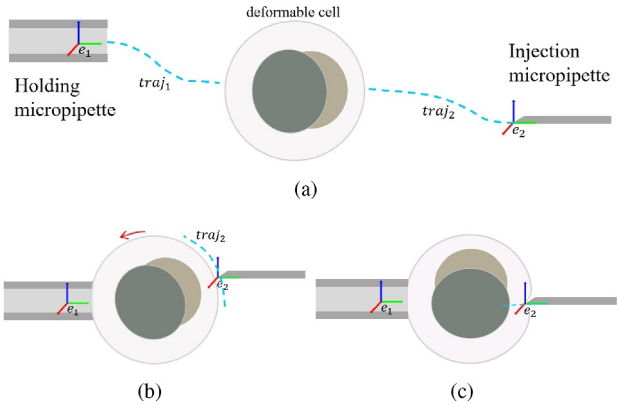


FIGURE 3 | Schematic diagram of cell manipulation. (a) First, the robot will control the left and right end-effectors to move along a defined trajectory to the vicinity of the cell on behalf of the manipulation. (b) The holding micropipette will capture the cells using negative pressure. To ensure the viability of the cell manipulation, the injection needle needs to first adjust the attitude of the cell. (c) The injection needle is punctured along the transverse axis of the cell towards the centre of the cell to complete the injection of the substance.

between the end-effectors and the cells, as well as the variations in the mask, can reflect the state of the robotic cell micromanipulation task. The detected states as well as the trajectories $\{e_{1,t},\ e_{2,t},\ e_{3,t},\ \omega_t,\ m_{c,t}\}_{t=1}^{T}$ are coupled with the time series dimension to obtain spatiotemporal state trajectories. MTO transforms video sequences into observation sequences containing multiple types of information. The observation sequence is denoted by $\{\xi_t\}_{t=1}^{T}$. MTO network training and inference are summarised as Algorithm 1.

2.1.1 | The Network Architecture

The backbone of the MTO network is ResNeXt. The region proposal network (RPN) is used to generate candidate target regions on the feature map. The RPN serves as the target detection branch, conducting bounding box regression and class classification to identify two end-effectors. The cell mask segmentation branch is based on RoIAlign. RoIAlign aligns regions of interest (RoI) of different sizes onto a fixed-size feature map. Masks for object instances are generated on each RoI to determine the object's area at the pixel level, used to detect cell deformation. The keypoint detection branch (keypoint head) is responsible for predicting a heatmap of keypoints within each RoI, used to identify the cell poses. The hourglass network is utilised to generate the heatmap of keypoints. By stacking these stages, the hourglass network learns spatial information and features from the image. Finally, a keypoint heatmap generation outputs a heatmap where each channel represents a keypoint.

2.1.2 | Loss Function for Multi-Task Observation

The total loss $\mathcal{L}_{total}$ of MTO consists of the weighted sum of the individual subtask loss functions

$$\mathcal{L}_{total} = \lambda_{obj} \cdot \mathcal{L}_{obj} + \lambda_{mask} \cdot \mathcal{L}_{mask} + \lambda_{key} \cdot \mathcal{L}_{key} \tag{1}$$

where $\mathcal{L}_{obj}$ is the object detection loss, $\mathcal{L}_{mask}$ is the instance segmentation loss. $\mathcal{L}_{key}$ is the keypoint detection loss and $\lambda_{obj}, \lambda_{mask}, \lambda_{key}$ are the weights for each task loss. $\mathcal{L}_{obj} = \mathcal{L}_{obj}^{bbox} + \mathcal{L}_{obj}^{class}$. The bounding box regression loss $\mathcal{L}_{obj}^{bbox}$

$$\mathcal{L}_{obj}^{bbox} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{Smooth}_{\mathrm{L1}}\left(B_{gt}^{i} - B_{pred}^{i}\right) \tag{2}$$

where N represents the total number of bounding boxes, B_{gt} refers to the ground truth bounding box and B_{pred} represents the predicted bounding box. The class classification loss $\mathcal{L}_{obj}^{class}$

$$\mathcal{L}_{obj}^{class} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log\left(p_{i,c}\right) \tag{3}$$

where N represents the total number of instances, C is the number of classes, $y_{i,c}$ is the ground truth class label for instance i and $p_{i,c}$ is the predicted probability of class c for instance i. The instance segmentation loss $\mathcal{L}_{mask}$

$$\mathcal{L}_{mask} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{H\times W} M_{gt}^{i,j}\log\left(M_{pred}^{i,j}\right) \tag{4}$$

where N is the total number of instances, $H \times W$ represents the pixel dimensions of the mask, $M_{gt}^{i,j}$ is the ground truth mask value and $M_{pred}^{i,j}$ is the predicted mask value. The keypoint detection loss $\mathcal{L}_{key}$

$$\mathcal{L}_{key} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{H\times W}\sum_{k=1}^{K}\left(H_{gt}^{i,j,k} - H_{pred}^{i,j,k}\right)^2 \tag{5}$$

where N is the total number of instances, K represents the number of keypoints, $H \times W$ represents the pixel dimensions of the heatmap, $H_{gt}^{i,j,k}$ is the ground truth heatmap value for keypoint k and $H_{pred}^{i,j,k}$ is the predicted heatmap value for keypoint k. The Gaussian distribution equation used to denote the heatmap values $H(x, y)$ at each pixel location (x, y) is

$$H(x,y) = \exp\left[-\frac{(x-\mu_x)^2 + (y-\mu_y)^2}{2\sigma^2}\right] \tag{6}$$

where μ_x and μ_y are the coordinates of the keypoint (the mean of the Gaussian distribution), and σ controls the variance of the Gaussian distribution.

ALGORITHM 1 | MTO network training and inference.

Input: Multi-task observation network *MTO*, dataset *D*;
1: **for** each epoch **do**
2: sample image, ground truth I, G from D;
3: predict $B_{pred}, p_c, M_{pred}, H_{pred}$ from $MTO(I)$;
4: calculate $\mathcal{L}_{obj}$ via (2) (3); ▷object detection
5: calculate $\mathcal{L}_{mask}$ via (4); ▷instance segmentation
6: calculate $\mathcal{L}_{key}$ via (5) (6); ▷keypoint detection
7: calculate $\mathcal{L}_{total}$ via (1); ▷total loss
8: update *MTO* with ADAM; ▷MTO training;

9: **end for**
10: **for** each frame in video **do**
11: $\xi_t \leftarrow MTO(image)$; ▷MTO inference
12: **end for**
Output: $\{\xi_t\}_{t=1}^{T}$

2.2 | Spatiotemporal Constraints in Expert Demonstrations

Spatiotemporal state trajectories are compressed, encoded and decoded by THMM to identify implicit constraints for micromanipulation tasks.

2.2.1 | THMM Theoretical Foundations

The observations may be arbitrary features defining environmental task variables, such as end-effectors position, velocity, acceleration and posture. The observation sequence is coupled to the unknown implicit state sequence which can be represented as $\{z_t\}_{t=1}^{T}$, $z_t \in \{1...K\}$ where the K hidden states form a discrete set. Different hidden states may correspond to different constraints in the robot micromanipulation task. And the hidden state follows a distribution of $z_t \sim \boldsymbol{\pi}_{z_{t-1}}$, where $\boldsymbol{\pi}_{z_{t-1}} \in \mathbb{R}^K$ is the last transition distribution with Π_i as the initial probability. The transfer from state i to state j can be denoted as $a_{i,j} = P(z_t = j | z_t = i)$, where $\boldsymbol{a} \in \mathbb{R}^{K \times K}$ is the transition matrix. Staying s consecutive time steps in state j and $p(s)$ has mean and standard deviation parameters denoted as $\left\{\mu_j^S, \Sigma_j^S\right\}$, which is estimated by Gaussian $\mathcal{N}\left(s|\mu_j^S, \Sigma_j^S\right)$. The observation feature ξ_t is derived from the distribution of state j, which is denoted by the multivariate Gaussian with mean and standard deviation parameters $\left\{\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right\}$. The set of overall parameter can be defined as $\Theta = \left\{\Pi_i, \{a_{i,m}\}_{m=1}^{K}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, \mu_i^S, \Sigma_i^S\right\}_{i=1}^{K}$. The infer constraints of the hidden state sequence $\{z_t\}_{t=1}^{T}$ from the observations should be modelled and encoded. The data generating distribution $p(\xi_t, z_t)$ is simplified for inferring the model structure $p(z_t|\xi_t)$ better, and the joint probability density of the hidden state sequence and observation $p(\xi_t, z_t)$ is learnt.

Hidden Markov models enhance a mixture model by incorporating latent states that evolve sequentially over time in demonstrations, thereby encapsulating spatiotemporal information. THMM can be described as a doubly stochastic process consisting of a sequence of hidden states and a sequence of observations/emissions. The utilisation of THMM for spatiotemporal encoding enables the handling of movements with varying durations, recurring patterns, movement options, as well as partial or unaligned demonstrations.

2.2.2 | THMM Parameters Learning Process

To infer and learn in THMM, the intermediary variables are leveraged as forward variable $\alpha_{t,i} = p(z_t = i, \xi_1 ... \xi_t | \theta)$, backward variable $\beta_{t,i} = p(\xi_{t+1} ... \xi_T | z_t = i, \theta)$, marginal of smoothed node $\gamma_{t,i} = p(z_t = i, \xi_1 ... \xi_T, \theta)$ and marginal of smoothed edge $\zeta_{t,i,j} = p(z_t = i, z_{t+1} = j | \xi_1 ... \xi_t, \theta)$.

$$\begin{aligned}
\alpha_{t,i} &= \left(\sum_{j=1}^{K} \alpha_{t-1,j}, a_{j,i}\right) \mathcal{N}(\xi_t | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \\
\beta_{t,i} &= \sum_{j=1}^{K} a_{j,i} \mathcal{N}\left(\xi_{t+1} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right) \beta_{t+1,j} \\
\gamma_{t,i} &= \frac{\alpha_{t,i} \beta_{t,i}}{\sum_{k=1}^{K} \alpha_{t,k} \beta_{t,k}} \\
\zeta_{t,i,j} &= \frac{\alpha_{t,i} \mathcal{N}\left(\xi_{t+1} | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right) \beta_{t+1,i}}{\sum_{k=1}^{K} \sum_{l=1}^{K} \alpha_{t,k} \mathcal{N}(\xi_{t+1} | \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l) \beta_{t+1,l}}
\end{aligned} \tag{7}$$

The number of demonstrations in a set is M and the complete log-likelihood expected by THMM is

$$Q\left(\theta, \theta^{old}\right) = E\left\{\sum_{m=1}^{M} \sum_{t=1}^{T} \log p(\xi_{m,t}, z_t | \theta) | \xi, \theta^{old}\right\} \tag{8}$$

which can be maximised by the expectation-maximisation (EM) method. By leveraging the EM algorithm, the parameters $\left\{\Pi_i, \{a_{i,m}\}_{m=1}^{K}, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right\}_{i=1}^{K}$ of THMM are estimated. And $\left\{\mu_i^S, \Sigma_i^S\right\}_{i=1}^{K}$ as duration parameters are estimated from the computed hidden state sequence of the observation sequence.

$$\begin{aligned}
Q\left(\theta, \theta^{old}\right) = &\sum_{i=1}^{K} \sum_{m=1}^{M} \gamma_{m,1,i} \log \Pi_i \\
&+ \sum_{m=1}^{M} \sum_{t=1}^{T} \sum_{i=1}^{K} \sum_{j=1}^{K} \zeta_{m,t,i,j} \log a_{i,j} \\
&+ \sum_{m=1}^{M} \sum_{t=1}^{T} \sum_{i=1}^{K} p\left(z_t = i | \xi_{m,t}, \theta^{old}\right) \log \mathcal{N}\left(\xi_{m,t} | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\right)
\end{aligned} \tag{9}$$

$$\begin{aligned}
&E - step : \gamma_{m,t,i} = \frac{\alpha_{t,i} \beta_{t,i}}{\sum_{k=1}^{K} \alpha_{t,k} \beta_{t,k}} \\
&M - step : \Pi_i \leftarrow \frac{\sum_{m=1}^{M} \gamma_{m,1,i}}{M} \\
&a_{i,j} \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m - 1} \zeta_{m,t,i,j}}{\sum_{m=1}^{M} \sum_{t=1}^{T_m - 1} \gamma_{m,t,i}} \\
&\boldsymbol{\mu_i} \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i} \xi_{m,t}}{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i}} \\
&\boldsymbol{\Sigma_i} \leftarrow \frac{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i} (\xi_{m,t} - \boldsymbol{\mu_i})(\xi_{m,t} - \boldsymbol{\mu_i})^T}{\sum_{m=1}^{M} \sum_{t=1}^{T_m} \gamma_{m,t,i}}
\end{aligned} \tag{10}$$

2.2.3 | Decoding for Obtaining Spatiotemporal Constraints

After the model parameters are estimated, for a given sequence of observations, the Viterbi algorithm can decode the most

likely sequence of hidden states. The decoding problem is denoted as

$$z_t^* = \arg\max_{z_1 \ldots z_T} p(\xi_1 \ldots \xi_T, z_1 \ldots z_T | \Theta) \tag{11}$$

Given the model parameters, computing the probability that the observation sequence is in the hidden state $z_t = i$ at the end of the sequence can be denoted as

$$\begin{aligned} h_{t,i} &= p(z_t | \xi_1 \ldots \xi_t) = \frac{\alpha_{t,i}}{\sum_{k=1}^{K} \alpha_{t,k}} \\ &= \frac{\pi_i \mathcal{N}(\xi_t | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\xi_t | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)} \end{aligned} \tag{12}$$

To predict the next time horizon states sequence $p(z_t, z_{t+1} \ldots z_{T_p} | \xi_1 \ldots \xi_t)$, deterministic sampling can be further implemented. Given observation sequence $\{\xi_1 \ldots \xi_t\}$ without the observation of next steps, the probability of ξ_t in state i is sampled by a forward variable $\alpha_{t,i}$, which is explicitly expressed as

$$\alpha_{t,i} = \sum_{j=1}^{K} \sum_{s=1}^{t-1} \alpha_{t-s,j} a_{j,i} \mathcal{N}(s | \boldsymbol{\mu}_i^S, \boldsymbol{\Sigma}_i^S) \tag{13}$$

The movement constraints sequence $\{t + 1, \ldots, N\}$ for N steps can be planned by $\alpha_{t,i}$.

$$\boldsymbol{z}_t = \{z_{t+1}, \ldots, z_N\} = \underset{i}{argmax} \alpha_{t,i} \tag{14}$$

$$\hat{\boldsymbol{\mu}}_t = \boldsymbol{\mu}_{z_t}, \ \hat{\boldsymbol{\Sigma}}_t = \boldsymbol{\Sigma}_{z_t} \tag{15}$$

The state space S_{THMM} consists of a set of unique states in the sequence of hidden states.

$$S_{THMM} = \{s_1, s_2, \ldots, s_K\} = \text{unique}(\{z_{t+1}, \ldots, z_N\}) \tag{16}$$

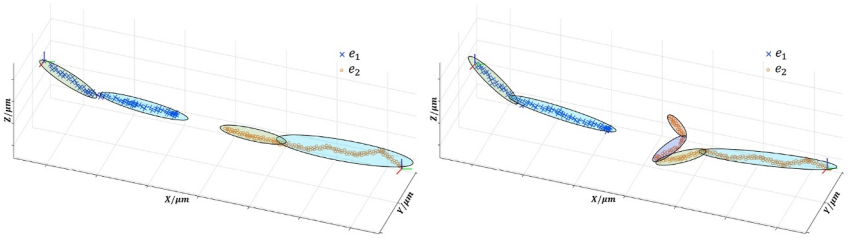
where K is the number of distinct hidden states in the trajectory. S_{THMM} was considered as spatiotemporal constraints, which was part of the state space and represented the potential states involved in the demonstration trajectory, as shown in Figure 4.

2.3 | Skill Learning by Demonstration and Exploration

ALGORITHM 2 | ILOSAC algorithm.

Input: critic θ, policy ϕ, replay buffer $\mathcal{D}$;
$\{\xi_t\}_{t=1}^{T} \leftarrow MTO(videos)$; ▷ multi-task observation
$S_{THMM} \leftarrow from\ (7) \sim (16)$; ▷ get constraints
for each iteration **do**
  for each environment step **do**
    $s_i \leftarrow MTO(image)$; ▷ observe state
    if $s_i \in S_{THMM}$ **then**
      $a_i \sim \pi_\phi(a_i | s_i)$; ▷ select action
      $execute(a_i)$;
      $s_{i+1} \sim p(s_{i+1} | s_i, a_i)$; ▷ observe next state
      if $s_{i+1} \notin S_{THMM}$ **then**
        $r_i \leftarrow r_i + \lambda_s^t \cdot r_{penalty}$;
        if $\lambda_s^t < \lambda_0$ **then**
          $\mathcal{D} \leftarrow (s_i, a_i, r_i, c_i, s_{i+1})$; ▷ store in buffer
          $S_{THMM} \leftarrow S_{THMM} \cup s_{i+1}$;
        end if
      else
        $\mathcal{D} \leftarrow (s_i, a_i, r_i, c_i, s_{i+1})$;
      end if
    end if
  end for
  for each gradient step **do**
    $\mathcal{B} \leftarrow \mathcal{D}$; ▷ randomly sample a batch of transition
    $\theta \leftarrow \theta - \lambda_Q \nabla_\theta \mathcal{L}_Q(\theta)$; ▷ update θ via (17)
    $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi \mathcal{L}_\pi(\phi)$; ▷ update ϕ via (20)
    $\overline{\theta} \leftarrow \tau\theta + (1 - \tau)\overline{\theta}$, $(0 < \tau \ll 1)$; ▷ update weights
    $\alpha \leftarrow \alpha - \lambda_\alpha \nabla_\alpha \mathcal{L}(\alpha)$; ▷ update α via (18)
  end for
end for
Output: ϕ

Robot micromanipulation falls under the category of continuous control problems. For control tasks in continuous action spaces, several reinforcement learning algorithms have been proposed, such as deep deterministic policy gradient (DDPG) [25] and soft actor-critic (SAC) [26] algorithms. These reinforcement learning algorithms have shown satisfactory performance in specific macro-level complex scenarios with multiple degrees of freedom, and they also bring a certain level



(a) (b)

FIGURE 4 | Conceptual diagram of the THMM for imitation learning. (a) and (b) are the trajectories generated by the two end-effectors of the cell manipulation robot as it moves through the cell puncture and cell posture adjustment, respectively. A Gaussian (shown as an ellipsoid) is used to encode the spatiotemporal constraints of the hidden state at different locations during robot manipulation.

of dexterity. However, in the context of robot micromanipulation, both the dexterity of robot operations and the safety of robot actions are crucial evaluation metrics. Therefore, to simultaneously address the safety and dexterity of micromanipulation, the ILOSAC integrated with MTO and THMM was proposed, which was summarised in Algorithm 2. This method not only enables the learning of dexterous manipulation strategies but also captures the spatiotemporal constraints of micromanipulation actions.

2.3.1 | ILOSAC Integrated With MTO and THMM

The ordinary Markov decision process (MDP) is extended to a spatiotemporal constraints Markov decision process (SCMPD), which can be represented as $(\mathcal{S}, \mathcal{A}, P, R, C)$. $\mathcal{S}$ and $\mathcal{A}$ represent the state and action space, respectively, R and C represent the reward function and cost function, respectively. P represents the transition probability from state s to state s' given action a, which is denoted as $p(s'|s,a)$. The micromanipulation robot is considered as an agent with two robot arms. The action a_i consists of moving direction d_i and micromanipulation speed v_i, which is denoted as $a_i = \left(d_i^l, \|v_i^l\|, d_i^r, \|v_i^r\|\right)$, where l and r denote the left and right micro-end-effectors, respectively. To make the motion smoother, set $d = 0.8d_i + 0.2d_{i-1}$.

The state, action, reward, loss and next state are stored as the experience data in a specific replay buffer set $\mathcal{D}$. The algorithm will periodically sample data from the experience replay pool based on whether the judgement conditions are met in order to improve data utilixation efficiency. The policy $\pi_\phi(a|s)$ is parameterised by a Gaussian function with mean $\mu_\phi(s)$ and covariance $\sigma_\phi(s)$ given by a deep neural network (DNN). The critic function $\mathcal{Q}_\theta(s,a)$ is defined by a DNN with parameter θ to estimate the expected return in state s taking action a. And the critic parameter θ is trained for minimising loss $\mathcal{L}_\mathcal{Q}(\theta)$.

$$\mathcal{L}_\mathcal{Q}(\theta) = E_{(s_i,a_i)\sim\mathcal{D}}\left\{\frac{1}{2}\left[\mathcal{Q}_\theta(s_i,a_i) - y_i\right]^2\right\} \tag{17}$$

where $\mathcal{Q}_\theta(s_i,a_i) = E_\pi\left[\hat{R}(\pi,\lambda,\alpha)|s_0 = s, a_0 = a\right]$ and $y_i = r(s_i, a_i) + \gamma\left(\mathcal{Q}_\theta(s_{i+1}, a_{i+1}) - \alpha\log\left(\pi_\phi(a_{i+1}|s_{t+1})\right)\right)^2$, γ is the reward discount factor and α is the temperature coefficient used to control the balance between exploration and exploitation.

$$\mathcal{L}_\alpha = E_{a_i\sim\pi_i}[-\alpha\log\pi_i(a_i|\pi_i) - \alpha\mathcal{H}_0] \tag{18}$$

The temperature coefficient α, as a hyperparameter, controls the emphasis of maximum entropy reinforcement learning (MERL) on the entropy. For tasks trained at different stages, each stage may have its own suitable value, which can be adjusted by the algorithm itself. This can be formulated as a constrained optimisation problem: maximising the expected return while maintaining the policy entropy above an entropy threshold $\mathcal{H}_0$.

A policy $\pi(a_i|s_i)$ of stochasticity is the function that maps from states s_i to the probability distribution over actions a_i. At each time point, the workspace under the microscope is parsed as the observation state s_i and gets an action a_i that is sampled using the mean μ_π and standard deviation Σ_π from the current policy π. The policy π_i is constrained by behaviour cloning policy π_z leveraging Kullback–Leibler divergence $D_{KL}(\pi_z\|\pi_i)$.

$$\min_{\pi_i} D_{KL}(\pi_z\|\pi_i) = \min_{\pi_i}\sum_j \pi_z(j)\log\left(\frac{\pi_z(j)}{\pi_i(j)}\right) \tag{19}$$

where $\pi_z(a_i|z_i,\xi_i)$ is a policy mapping function constructed using state sequences and action sequences. And the policy parameters $\phi = \left\{\phi_\mu, \phi_\sigma\right\}$ can be optimised by minimising the loss function.

$$\begin{aligned}\mathcal{L}_\pi(\phi) &= E_{s_i\sim\mathcal{D}}\left[\alpha\log\pi_\phi(a_i|s_i) - \mathcal{Q}_\theta(s_i,a_i)\right] \\ &\quad + \lambda_{KL}^t \cdot D_{KL}(\pi_z\|\pi_\phi)\end{aligned} \tag{20}$$

The detail of ILOSAC is shown in Algorithm 2. Network *MTO* constantly obtains the state information s_i from the environment. In the initial phase, the spatiotemporal constraint S_{THMM} obtained from THMM will limit the search space of ILOSAC to bootstrap skill learning by demonstration. Actions a_i that only sampled within the constrained state space $s_i \in S_{THMM}$ are allowed to be executed. If an unreasonable state occurs, a penalty is applied to avoid potential damage to cells. The exploration capability of ILOSAC is controlled by λ_0 and λ_s $(\lambda_0 < \lambda_s < 1)$. The KL-divergence constraint forces the policy to stay close to the demonstration distribution, reducing initial exploration randomness. As the number of episodes increases, λ_{KL}^t $(\lambda_{KL} < 1)$ will decrease and λ_s^t will be smaller than λ_0 and the state that do not satisfy the constraint will be merged $S_{THMM} \cup s_{i+1}$. The state search space of the policy is gradually and dynamically released. The constraints are slowly weakened and ILOSAC gradually learns new micromanipulation skills by exploration. In the exploration process, the primary means of supervising and constraining the agent's actions relies on the reward function.

2.3.2 | Exploration Reward Function

To ensure the balance of the reward function, avoid reward sparsity and reward misguidance and promote effective learning and optimised manipulation strategies for the robot, a comprehensive reward function is designed. Goal achievement reward: When the robot moves the end-effectors to the specified cell location and performs the manipulation, a higher reward is given. Action penalty: Negative rewards are given when the robot performs unreasonable actions or causes damage to the cells. Negative rewards are given when the robot moves the end-effectors to the wrong position. Intervention degree penalty: Negative rewards are given when the robot causes excessive interference to the cells or when the degree of cell deformation is significant. Time efficiency: Cell micromanipulation needs to be completed within a certain time frame. In order to minimise cell damage from robotic cell manipulation and to ensure the end-effector is correctly positioned close to the cell, r_{base} was designed. And to ensure that the robot is able to perform specific tasks (cell inject, cell posture adjustment and composite

micromanipulation), a subtask reward function r_{task} was designed. The reward function $r_i = r_{base,i} + r_{task,i}$ is designed as follows:

$$\begin{aligned} r_{base,i} = & -\|e_{2,i} - m_{c,i}\|^2 - \|e_{1,i} - m_{c,i}\|^2 \\ & - \mathrm{std}\left(\{m_{c,t}\}_{t=0}^{i}\right) - \|m_{c,i} - m_{c,i-1}\|^2 \\ & - 0.1N_{steps} \end{aligned} \tag{21}$$

$$\begin{aligned} r_{inject,i} &= -2\|e_{2,i} - e_{3,i}\|^2 \\ r_{adjust,i} &= -2\|\omega_{m_{c,i}} - \omega_{desire}\|^2 \end{aligned} \tag{22}$$

$$r_{composite,i} = \begin{cases} -c_1 \\ -3\|\omega_{m_{c,i}} - \omega_{desire}\|^2 \\ -c_2 \\ -3\|e_{2,i} - e_{3,i}\|^2 \end{cases} \tag{23}$$

where $-\|e_{2,i} - m_{c,i}\|^2$ and $-\|e_{1,i} - m_{c,i}\|^2$ constrain the injection needle and holding micropipette to be close to the cell. $-std\left(\{m_{c,t}\}_{t=0}^{i}\right)$ constrains the deformation of the cell so that the shape deformation during manipulation is minimised. $-\|m_{c,i} - m_{c,i-1}\|^2$ prevents the cell from producing a large deformation in a short period of time. $-0.1N_{steps}$ prevents excessive operation time. $-\|e_{2,i} - e_{3,i}\|^2$ prompts the injection needle to reach the centre of the cell for the puncture micromanipulation. $-\|\omega_{m_{c,i}} - \omega_{desire}\|^2$ prompts the injection needle to rub the cell to accomplish postural adjustment. $r_{composite,i}$ prompts the injection needle to complete the cell posturing and the cell puncture micromanipulation sequentially. If the system attempts a puncture before adjusting the orientation, $r_{composite,i} = -c_1(c_1 = 9)$. When the cell's orientation matches the target, $r_{composite,i} = -c_2(c_2 = 3)$.

3 | Experiment

In order to validate the effectiveness of the proposed framework, we tested the performance of the MTO as well as the ILOSAC algorithms, respectively. Finally, the whole framework is tested in a real robot manipulation scenario. We evaluate the performance of ILOSAC compared with the advanced imitation learning framework as well as other state-of-the-art deep RL algorithms. Next, we perform a self-comparison study to evaluate the contribution of imitation constraint optimisation components on the algorithm performance. Finally, the robot performed cell manipulation in a real environment, which validated the effectiveness of the proposed framework.

3.1 | Experiment Setup

Figure 5 depicted the multi-degree-of-freedom microscopic manipulation robotic system. The kinematic part of the system consists of two robotic arms with three degrees of freedom. The vision part is mainly composed of a CMOS camera, an inverted microscope system that automatically switches the optical magnification. The system consists of an inverted microscope, IXplore Standard (OLYMPUS, Japan) equipped with an XY motorised carrier stage and two microscope-manipulation robotic arms, TransferMan 4M (EPPENDORF, Germany) with three degrees of freedom. The maximum magnification of the

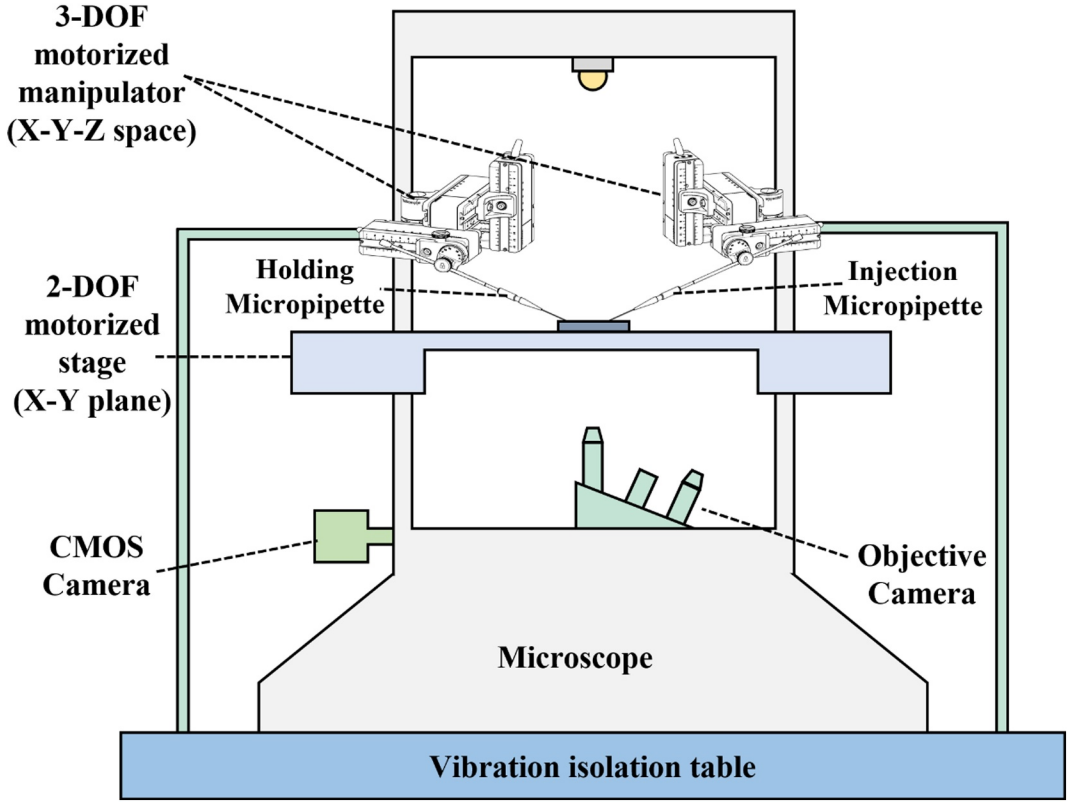


FIGURE 5 | Schematic diagram of the robot hardware system.

microscope is 400x through the eyepiece and objective lens. The robotic arms have a motion resolution of 0.05 μm. A camera is attached to the inverted microscope to capture image information. We integrated the software and control part of the system into the robot operating system (ROS) Melodic, which runs on a desktop computer with Intel Core i9-10980Xe CPU and NVIDIA TITAN Xp in Ubuntu 18.04.

3.2 | MTO Performance Characterisation

The loss function convergence curve of the MTO network is displayed in Figure 6. To verify the accuracy and robustness of MTO for multi-task observations, we tested the original dataset as well as the data with added noise. Figure 7a,c demonstrates that the MTO algorithm can effectively accomplish end-effectors target detection, cell instance segmentation and cell pose recognition. (b) (d) Reveals that the MTO algorithm is still able to accomplish multitasking observations when dealing with data with strong noise.

As shown in Table 1, the target detection mAP (0.5:0.95) for processing the original dataset as well as the dataset containing noise is 0.928 and 0.911, respectively, with a variation of 1.83%. Examples of segmentation mAP (0.5:0.95) are 0.863 and 0.852 with a variation of 1.27%. Angle recognition errors were 1.47° and 1.49°, respectively, with a variation of 1.36%. The experimental results demonstrate the accuracy and robustness of MTO for multi-task observation.

3.3 | Comparison With Benchmark Algorithms

3.3.1 | Experiment Result

We performed four continuous control tasks, which included two end-effectors approaching the manipulated object, cell puncture, cell posture adjustment and a composite task of adjustment followed by puncture. There are 10 demonstration videos for each task. Figure 8 illustrates the learning curves of all the algorithms, the *x*-axis represents the number of episodes and the *y*-axis represents the average return obtained by the agent during the interactions.

- Base task: controlling two end-effectors close to the manipulated object.
- Injection task: controlling two end-effectors for cell injection, where the holding micropipette is required to suction the cells and the injection needle to perform cell puncture.
- Adjustment task: controlling the two end-effectors for the cell posture adjustment task.
- Composite task: controlling two end-effectors for a combined cell posture adjustment and puncture task.

The four tasks varied in complexity and difficulty, with the base task being the simplest, followed by cell puncturing and cell pose adjustment tasks of moderate complexity and the composite task being the most complex. Table 2 presents the average final rewards obtained by different algorithms over 5 runs.

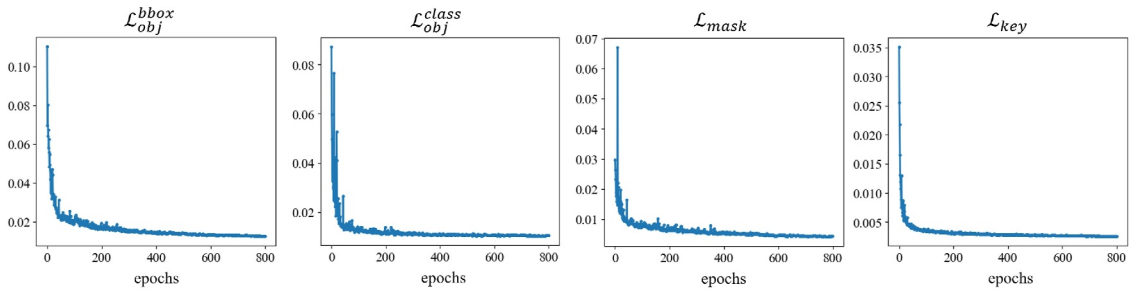


FIGURE 6 | Convergence curves of the four losses of the MTO network on the test set.

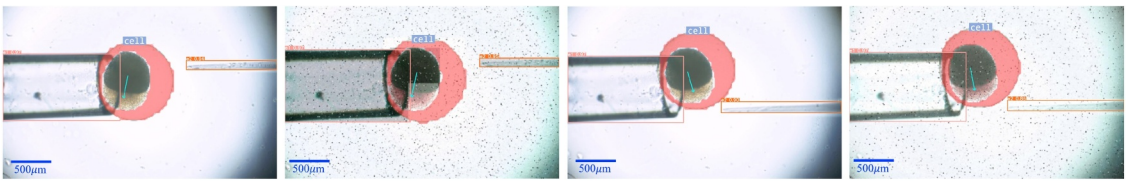


FIGURE 7 | MTO multi-task output results with robustness testing. (a, c) Output results for workspace observation of real cellular manipulation scenarios. (b, d) Output results for workspace observation with noise interference.

TABLE 1 | MTO robustness validation.

Task	Detect (mAP) ↑	Segment (mAP) ↑	Pose error (°) ↓
No noise	0.928	0.863	1.47
With noise	0.911	0.852	1.49
Variation	1.83%	1.27%	1.36%

3.3.2 | Performance Analysis in Each Task

In the base task, as shown in Figure 8a, there were small differences in learning rates and stability among the algorithms. All the algorithms achieve high returns with low variance convergence, and ILOSAC achieves the highest return of −0.32.

In the injection tasks, as shown in Figure 8b, ILOSAC demonstrates the clear advantages. Since the SAC and DDPG algorithms have unconstrained policy gradients for policy updating, the learning curves of both have strong oscillations. Compared to the other methods, ILOSAC has the fastest learning rate and achieves the highest average return of −0.81.

In the adjustment tasks, as shown in Figure 8c, guided by imitation learning, all four imitation learning frameworks yielded high returns in a short number of interactions. However, without the aid of imitative learning, SAC and DDPG of starting from scratch makes it difficult to learn effective operational skills in a short number of interactions. The average return of ILOSAC significantly outperforms other methods.

For the composite task, as shown in Figure 8d, the ILOSAC algorithm still obtains the highest average return. The final returns of DDPG and SAC algorithms are low and have large variance due to the high complexity of the task. ILOSAC outperforms DDPG and SAC due to its rapid acquisition of skills through the application of spatiotemporal constraints. Moreover, ILOSAC is capable of learning dexterous skill by weakening constraints and increasing space for exploration. Consequently, ILOSAC achieves higher rewards compared to current imitation learning methods SOIL, DAPG and DDPGfD.

The above results and analysis demonstrate that the learning capability of the proposed framework significantly outperforms the benchmark algorithms in robot micromanipulation skill learning.

3.4 | Ablation Studies

To further validate the significant contribution of the THMM and imitation optimisation component in the framework, we compare the performance difference between ILOSAC and SAC. As shown in Figure 9 and Table 3, adding spatiotemporal constraints through imitation learning significantly improves the learning ability of the ILOSAC algorithm. Ablation experiments demonstrate the performance of the proposed framework during robot micromanipulation. By introducing the THMM and imitation optimisation component, the ILOSAC algorithm improves the performance of the four tasks by 50.7%, 28.3%, 40.8% and 68.6%, respectively, compared with the SAC algorithm. The THMM and imitation optimisation component combines expert skill with an autonomous exploration of the algorithm by minimising the KL divergence of the policy distribution and the demonstration distribution, achieving a significant performance improvement over fully exploratory types of algorithms. These results demonstrate the effectiveness of the proposed framework.

3.5 | Manipulation Skills Validation

3.5.1 | ILOSAC Compared With IL and RL Methods

In a real microscopic view environment, as shown in Figure 10, we conducted the micromanipulation experiments using the agent to control the injection needle and the holding needle. The proposed multi-task observation network is utilised to provide the information feedback of vision and simultaneously acquire the position information of the two

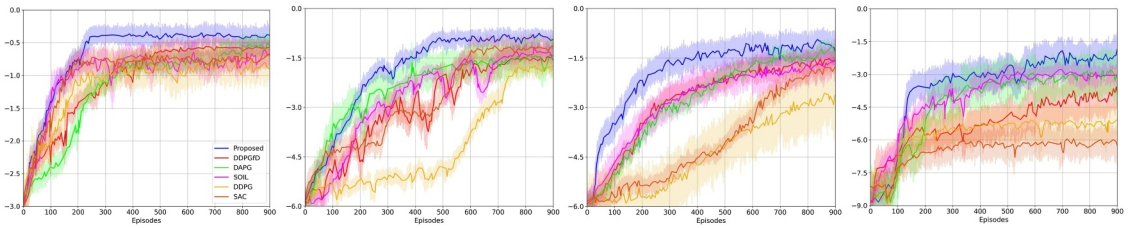


FIGURE 8 | Schematic diagram of the average reward return versus the number of iterations for different algorithms on different tasks. (a) Results of base task. (b) Results of injection task. (c) Results of adjustment task. (d) Results of composite task.

TABLE 2 | Average final return.

Method	Base ↑	Injection ↑	Adjustment ↑	Composite ↑
DDPG [25]	−0.73 ± 0.16	−1.65 ± 0.13	−2.63 ± 0.74	−5.14 ± 0.49
SAC [26]	−0.63 ± 0.09	−1.13 ± 0.22	−1.81 ± 0.15	−6.03 ± 0.42
SOIL [20]	−0.57 ± 0.13	−1.27 ± 0.16	−1.72 ± 0.31	−2.95 ± 0.26
DDPGfD [18]	−0.51 ± 0.04	−1.46 ± 0.21	−1.65 ± 0.18	−3.56 ± 0.54
DAPG [19]	−0.47 ± 0.08	−1.42 ± 0.32	−1.29 ± 0.11	−2.79 ± 1.06
Proposed	**−0.32 ± 0.06**	**−0.81 ± 0.13**	**−1.07 ± 0.24**	**−1.89 ± 0.51**

Note: ± corresponds to a single standard deviation over five runs. Bold values mean better result.

end-effectors, as well as the state information of the cell. As shown in Figure 10a, under the microscope, the cell is captured by the holding micropipette on the left side through negative pressure, and the injection needle on the right side contacts the cell along a certain trajectory and rubs the cell, which ultimately completes the dexterous manipulation of the cell's attitude adjustment. During the process of attitude adjustment, the cell did not undergo obvious deformation, which indicated that the robot did not cause obvious damage to the cell. The trajectory of the injection needle movement is shown in the light red part. As shown in Figure 10b, the cell was captured by the holding micropipette on the left side, and the injection needle on the right side moved slowly along the trajectory pointing towards the cell internal measurement and finally inserted into the cell to complete the cell puncture manipulation.

To demonstrate the superiority and task generalisation of the proposed method, validation experiments are made for each task in a real environment and the success rate is recorded. Test cases were active zebrafish embryonic cell with a diameter of $700 \sim 1000\,\mu m$. Parameters: The diameter of the holding micropipette was 800 μm and the diameter of the injection needle was 30 μm. The number of experiments for each algorithm was 100. Evaluation metrics: Whether the holding micropipette captures cells or not. Whether the posture adjustment angle is less than 10°. Whether the injection needle penetrates the cell. Whether the overall operation time is less than 10 s. Whether the cell or the end-effector is damaged. Success was defined as the cell being adjusted to the specified attitude and completing the puncture. Figure 11 illustrates the common task failure states. Catching fail occurs mainly because the end-effector holding micropipette does not successfully approach and aspirate zebrafish embryonic cells. The positioning error occurs when the trajectory of the end-effector holding micropipette deviates, resulting in a positioning error, and the trajectory of the injection needle does not allow for further puncture or postural adjustment. The main reason for cell damaged is that the relative trajectory of the end-effector injection needle and the holding micropipette extrudes on the cells, resulting in cell destruction. End-effector damaged is mainly due to the deviation of the trajectory of the end-effector

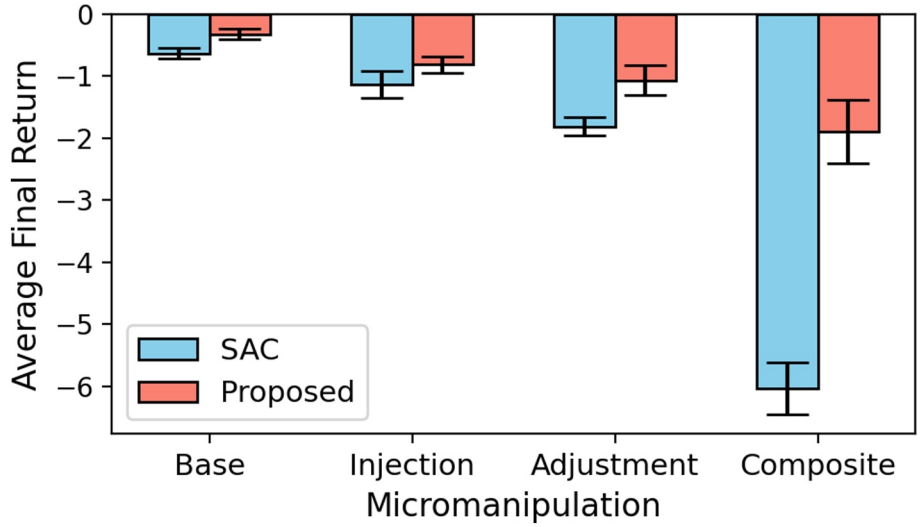
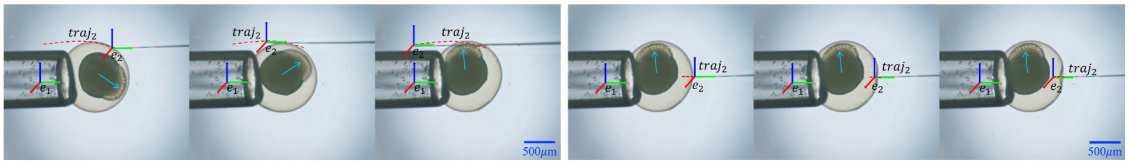


FIGURE 9 | Schematic diagram comparing the average reward returns of the ILOSAC and SAC algorithms. THMM and imitation optimisation component significantly improve the learning ability of the ILOSAC algorithm. Ablation experiments demonstrate the performance of the proposed framework during robot micromanipulation.

TABLE 3 | Performance improvement in ablation experiments.

Task	Base	Injection	Adjustment	Composite
Improvement	50.7%	28.3%	40.8%	68.6%



(a) (b)

FIGURE 10 | The ILOSAC algorithm controls two end-effectors for cell manipulation. (a) Cell attitude adjustment. The holding micropipette sucks the cell and the injection needle is rubbed along the trajectory. (b) Cell puncture. The holding micropipette sucks the cell and the injection needle is inserted into the cell interior along the trajectory.

needle, resulting in a large bending or even fracture of the needle after contact with the cell, causing damage to the needle. The aforementioned issues can be mitigated by increasing the amount of demonstration data and incorporating safe action constraints. DDPG and SAC often fail to capture cells and destroy cells. SOIL is prone to position errors during cell manipulation. In DAPG and DDPGfD, the injection needle is easily damaged during cell puncture. As shown in Table 4, the proposed framework has the highest success rate on multiple tasks.

To further evaluate the adaptability of the proposed method, its robustness against fluid-induced disturbances was tested. A flow control device was employed to regulate the solution flow rate in the Petri dish to 100 μm/s. As shown in Figure 12, under conditions without fluidic disturbances, the success rates for the base, inject, adjust and composite tasks were 97%, 86%, 77% and 68%, respectively. Under dynamic fluid interference at a flow rate of 100 μm/s, the success rates slightly decreased to 95%, 85%, 74% and 63%, respectively. The effect of fluid perturbation is mainly to slightly change the position of the cells. Although a marginal performance drop was observed, the proposed method still maintained relatively high success rates compared to other approaches.

3.5.2 | ILOSAC Compared With Conventional Methods

To further validate the superiority of the framework, we compare the manipulation strategy trained by the proposed framework with conventional methods and manual remote control (3 years of experience). The results of the quantitative experiments are exhibited in Table 5, and Figure 13. Smaller cell deformations with shorter manipulation times indicate safer as well as more dexterous manipulation strategies. As shown in Figure 13a, the angle changes are smoother during the robot's autonomous adjustment of the cell's posture. As shown in Table 5, the proposed method takes less time (2.72 s) compared to PID [27] and optimal control [6]. When manually controlling the lever, the operating lever needs to be reset and adjusted several times, so the attitude change is discontinuous and inefficient. As shown in Figure 13b, the robot autonomously performed cell injection with a shorter time and small cell

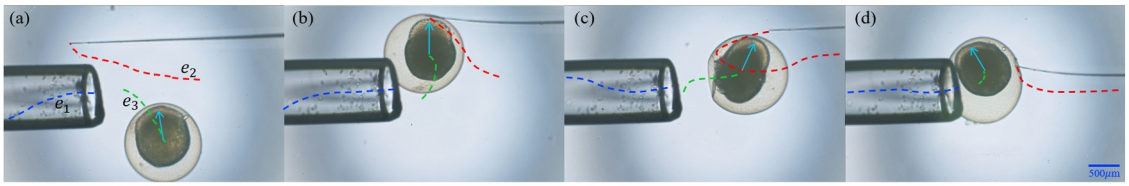


FIGURE 11 | Four kinds of cell manipulation failures.

TABLE 4 | The proposed method is compared with other methods in terms of success rate of micromanipulation tasks.

	Success rate (%) ↑			
Method	**Base**	**Injection**	**Adjustment**	**Composite**
DDPG [25]	77	56	38	19
SAC [26]	74	61	45	21
SOIL [20]	83	69	58	42
DDPGfD [18]	85	75	66	47
DAPG [19]	89	81	63	54
Proposed	**97**	**86**	**77**	**68**

Note: Bold values mean better result.

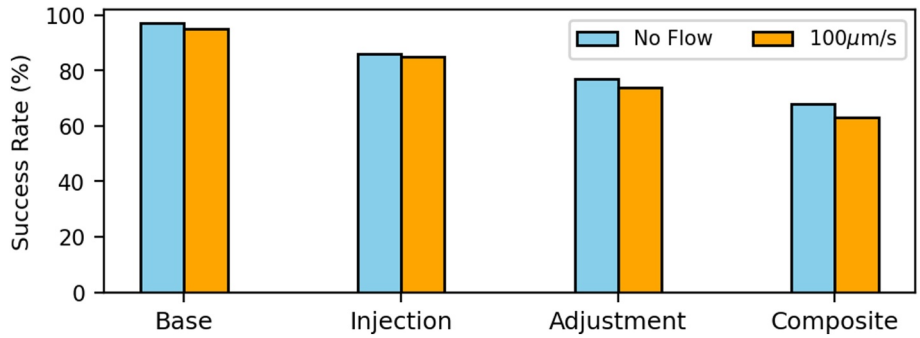
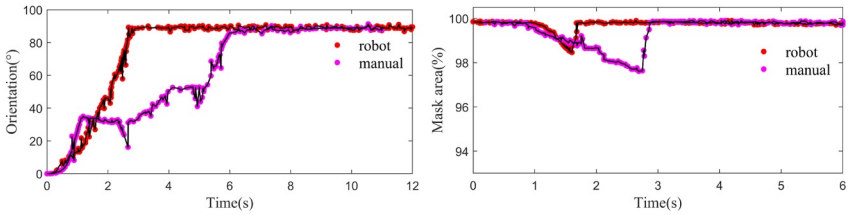


FIGURE 12 | Comparison of success rates with and without fluid interference.

TABLE 5 | Performance comparison of the proposed method with manual remote control and conventional methods.

Task	Injection (cell deformation)	Orientation adjustment (time)
Manual control	2.39%	> 6 s
PID control [27]	/	> 6.8 s
Optimal control [6]	/	3∼4 s
Proposed	1.54%	2.72 s



(a) (b)

FIGURE 13 | Comparison of autonomous robot manipulation and manual teleoperation control. (a) Cell attitude adjustment manipulation, which utilises two end-effectors to rotate the cell to a specified angle. (b) Cell injection manipulation, change of the cell mask area during cell injection by the micropipette.

deformation. Since it is difficult to ensure a stable axial movement speed when the manual control lever performs puncture, and the lever needs to be reset and adjusted after moving a certain distance, there is a large deformation and time consumption of the cells during the manipulation. The deformation of robotic autonomous manipulation and remote joystick control manipulation were 1.54% and 2.39%, respectively. This indicates that the proposed method causes less damage to the cells compared to manual control.

4 | Conclusion

An imitation learning framework inspired by the human learning process was proposed that enables robots to learn dexterous micromanipulation skills from videos. The MTO network is designed to obtain the spatiotemporal trajectories. The spatiotemporal constraints of the robot's actions are obtained by the THMM. To simultaneously address the safety and dexterity of robot micromanipulation, the ICOSAC algorithm was proposed in which the robot can perform skill learning by demonstration and exploration. The proposed method is capable of performing complex cell manipulation tasks in a realistic physical environment and outperforms current methods, which is expected to facilitate the development of robot skill learning. Expanding task dimensions to accommodate micromanipulation tasks with larger state spaces, more complex actions and higher levels of difficulty is part of our future research direction. The task dimensions can be expanded in the future by increasing the number of hidden states to capture finer-grained skill of manipulation. Low-dimensional representations of high-dimensional visual observations can be learnt to reduce the input space of the policy network, thereby addressing larger state spaces. Continuous actions can be decomposed and optimised step by step through the introduction of independent policy networks. A progressive learning strategy can also be adopted, transitioning from simple to complex actions by dynamically adjusting exploration parameters.

Funding

This work was supported in part with the General Programme of the National Natural Science Foundation of China (Grant 62576312), the Key Research and Development Program of Zhejiang Province (Grant 2025C01132) and the Shandong Province Key R&D Plan Project (Grant 2022LZGC020).

Conflicts of Interest

The authors declare no conflicts of interest..

Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

1. Y. Zhang, B. K. Chen, X. Liu, and Y. Sun, "Autonomous Robotic Pick-and-Place of Microobjects," *IEEE Transactions on Robotics* 26, no. 1 (2009): 200–207, https://doi.org/10.1109/TRO.2009.2034831.

2. L. Zheng, Y. Jia, D. Dong, et al., "3d Navigation Control of Untethered Magnetic Microrobot in Centimeter-Scale Workspace Based on Field-of-View Tracking Scheme," *IEEE Transactions on Robotics* 38, no. 3 (2021): 1583–1598, https://doi.org/10.1109/tro.2021.3118205.

3. S. Zhuang, C. Dai, G. Shan, C. Ru, Z. Zhang, and Y. Sun, "Robotic Rotational Positioning of End-Effectors for Micromanipulation," *IEEE Transactions on Robotics* 38, no. 4 (2022): 2251–2261, https://doi.org/10.1109/tro.2022.3142671.

4. C. Dai, G. Shan, H. Liu, C. Ru, and Yu Sun, "Robotic Manipulation of Sperm as a Deformable Linear Object," *IEEE Transactions on Robotics* 38, no. 5 (2022): 2799–2811, https://doi.org/10.1109/tro.2022.3158200.

5. Y. Zhang, X. Guo, Q. Wang, et al., "Automated Dissection of Intact Single Cell From Tissue Using Robotic Micromanipulation System,"

IEEE Robotics and Automation Letters 8, no. 8 (2023): 4705–4712, https://doi.org/10.1109/lra.2023.3287364.

6. C. Dai, Z. Zhang, Y. Lu, et al., "Robotic Manipulation of Deformable Cells for Orientation Control," *IEEE Transactions on Robotics* 36, no. 1 (2019): 271–283, https://doi.org/10.1109/tro.2019.2946746.

7. S. Hu and D. Sun, "Automatic Transportation of Biological Cells With a Robot-Tweezer Manipulation System," *International Journal of Robotics Research* 30, no. 14 (2011): 1681–1694, https://doi.org/10.1177/0278364911413479.

8. X. Liu, K. Kim, Y. Zhang, and Yu Sun, "Nanonewton Force Sensing and Control in Microrobotic Cell Manipulation," *International Journal of Robotics Research* 28, no. 8 (2009): 1065–1076, https://doi.org/10.1177/0278364909340212.

9. S. Permana, E. Grant, G. M. Walker, and J. A. Yoder, "A Review of Automated Microinjection Systems for Single Cells in the Embryogenesis Stage," *IEEE* 21, no. 5 (2016): 2391–2404, https://doi.org/10.1109/tmech.2016.2574871.

10. M. Xie, J. K. Mills, Y. Wang, M. Mahmoodi, and D. Sun, "Automated Translational and Rotational Control of Biological Cells With a Robot-Aided Optical Tweezers Manipulation System," *IEEE Transactions on Automation Science and Engineering* 13, no. 2 (2015): 543–551, https://doi.org/10.1109/tase.2015.2411271.

11. S. Yang and K. W. C. Lai, "A Novel Micropipette Robot for Cell Manipulation Based on Dielectrophoresis and Electroosmotic Vortex," in *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (IEEE, 2021), 24–29.

12. S. Schaal, A. Ijspeert, and A. Billard, "Computational Approaches to Motor Learning by Imitation," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* 358, no. 1431 (2003): 537–547, https://doi.org/10.1098/rstb.2002.1258.

13. Y. Ma, De Xu, and F. Qin, "Efficient Insertion Control for Precision Assembly Based on Demonstration Learning and Reinforcement Learning," *IEEE Transactions on Industrial Informatics* 17, no. 7 (2020): 4492–4502, https://doi.org/10.1109/tii.2020.3020065.

14. H. Su, W. Qi, Y. Hu, H. R. Karimi, G. Ferrigno, and E. De Momi, "An Incremental Learning Framework for Human-Like Redundancy Optimization of Anthropomorphic Manipulators," *IEEE Transactions on Industrial Informatics* 18, no. 3 (2020): 1864–1872, https://doi.org/10.1109/tii.2020.3036693.

15. L. Chen, Y. Wang, Z. Miao, et al., "Transformer-Based Imitative Reinforcement Learning for Multi-Robot Path Planning," *IEEE Transactions on Industrial Informatics* 19, no. 10 (2023): 10233–10243, https://doi.org/10.1109/tii.2023.3240585.

16. T. Hester, M. Vecerik, O. Pietquin, et al., "Deep q-Learning From Demonstrations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32, no. 1 (2018), https://doi.org/10.1609/aaai.v32i1.11757.

17. A. Nurbayeva, A. Shintemirov, and M. Rubagotti, "Deep Imitation Learning of Nonlinear Model Predictive Control Laws for Safe Physical Human-Robot Interaction," *IEEE Transactions on Industrial Informatics* 19, no. 7 (2022): 8384–8395, https://doi.org/10.1109/TII.2022.3217833.

18. M. Vecerik, Todd H., J. Scholz, et al., "Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems With Sparse Rewards," arXiv Preprint arXiv:1707.08817 (2017).

19. A. Rajeswaran, V. Kumar, A. Gupta, et al., "Learning Complex Dexterous Manipulation With Deep Reinforcement Learning and Demonstrations," in *Proceedings of Robotics: Science and Systems* (2018).

20. I. Radosavovic, X. Wang, L. Pinto, and J. Malik, "State-Only Imitation Learning for Dexterous Manipulation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2021), 7865–7871.

21. M. B. Hafez and S. Wermter, "Continual Robot Learning Using Self-Supervised Task Inference," *IEEE Transactions on Cognitive and Developmental Systems* 16, no. 3 (2024): 947–960, https://doi.org/10.1109/TCDS.2023.3315513.

22. W. Wan, Y. Zhu, R. Shah, and Y. Zhu, "Lotus: Continual Imitation Learning for Robot Manipulation Through Unsupervised Skill Discovery," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, https://doi.org/10.1109/ICRA57147.2024.10611129.

23. R. Mori, T. Aoyama, T. Kobayashi, K. Sakamoto, M. Takeuchi, and Y. Hasegawa, "Real-Time Spatiotemporal Assistance for Micromanipulation Using Imitation Learning," *IEEE Robotics and Automation Letters* 9, no. 4 (2024): 3506–3513, https://doi.org/10.1109/lra.2024.3366011.

24. A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation Learning: A Survey of Learning Methods," *ACM Computing Surveys* 50, no. 2 (2017): 1–35, https://doi.org/10.1145/3054912.

25. T. P. Lillicrap, J. J. Hunt, A. Pritzel, et al., "Continuous Control With Deep Reinforcement Learning," arXiv Preprint arXiv:1509.02971 (2015).

26. T. Haarnoja, A. Zhou, K. Hartikainen, et al., "Soft actor-critic Algorithms and Applications," arXiv Preprint arXiv:1812.05905 (2018).

27. C. Dai, G. Shan, X. Liu, C. Ru, L. Xin, and Yu Sun, "Automated Orientation Control of Motile Deformable Cells," *IEEE Transactions on Automation Science and Engineering* 20, no. 3 (2022): 2126–2134, https://doi.org/10.1109/tase.2022.3194845.