RM-RL: Role-Model Reinforcement Learning for Precise Robot Manipulation

Xiangyu Chen, Chuhao Zhou, Yuxi Liu, and Jianfei Yang[†]

Abstract— **Precise robot manipulation is critical for fine-grained applications such as chemical and biological experiments, where even small errors (e.g., reagent spillage) can invalidate an entire task. Existing approaches often rely on pre-collected expert demonstrations and train policies via imitation learning (IL) or offline reinforcement learning (RL). However, obtaining high-quality demonstrations for precision tasks is difficult and time-consuming, while offline RL commonly suffers from distribution shifts and low data efficiency. We introduce a Role-Model Reinforcement Learning (RM-RL) framework that unifies online and offline training in real-world environments. The key idea is a role-model strategy that automatically generates labels for online training data using approximately optimal actions, eliminating the need for human demonstrations. RM-RL reformulates policy learning as supervised training, reducing instability from distribution mismatch and improving efficiency. A hybrid training scheme further leverages online role-model data for offline reuse, enhancing data efficiency through repeated sampling. Extensive experiments show that RM-RL converges faster and more stably than existing RL methods, yielding significant gains in real-world manipulation: 53% improvement in translation accuracy and 20% in rotation accuracy. Finally, we demonstrate the successful execution of a challenging task, precisely placing a cell plate onto a shelf, highlighting the framework's effectiveness where prior methods fail.**

I. INTRODUCTION

Autonomous robotic manipulation has shown promising results in real-world tasks, such as folding clothes and performing household manipulations [1]–[3]. Beyond these general-purpose applications, growing attention has shifted toward high-precision manipulation, particularly in delicate biological and chemical experiments [4], [5]. In such domains, success hinges on the robot's ability to execute actions with sub-millimeter to millimeter-level accuracy, as even minor deviations can compromise the entire experiment.

Two primary paradigms have been widely adopted for robotic policy learning: Imitation Learning (IL) and Reinforcement Learning (RL). Imitation Learning (IL) [6], [7] provides an effective way to acquire policies from expert demonstrations. However, in high-precision tasks, even human operators struggle to deliver consistent millimeter-level accuracy through teleoperation, making the collection of demonstrations slow and costly. Learning from such limited data often leads to overfitting to the training distribution and poor generalization in real-world scenarios. In contrast, Reinforcement Learning (RL) enables policies to autonomously explore and optimize through trial and error, thereby eliminating the need for demonstrations. Yet, most RL methods are developed and validated in simulators [8]–[10], which fail to capture the subtle but critical errors in fine-grained real-world manipulation due to the persistent sim-to-real gap. This motivates our goal: to enable efficient training of robotic policies with high-precision manipulation skills directly in the physical world.

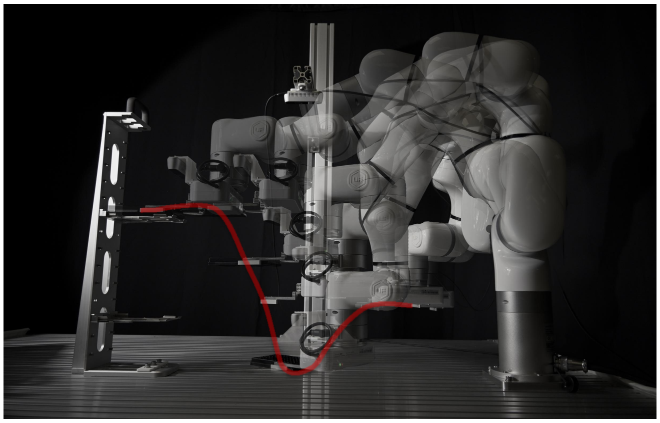

Fig. 1: The Ufactory X-ARM 6 autonomously executes a pick-and-place task, transferring a cell plate to the designated shelf. The red curve illustrates the trajectory of the end-effector.

However, applying online RL directly in the real world is far from straightforward. In existing online RL, policies are typically updated directly from real-world reward signals. This process is highly data-inefficient, as each interaction can only be used once, and data collection in the physical world is inherently slow. Attempts to improve efficiency with replay buffers [11]–[13] partially mitigate this issue but introduce a second challenge: distribution shift [14]–[16]. Because stored data is generated by outdated policies, the mismatch between past experiences and the current policy often destabilizes optimization and impedes convergence. These limitations lead us to the central question of this work: **Can we design a real-world RL framework that achieves both high data efficiency and training efficiency for precise robotic manipulation?**

> *"When three people walk together, there must be a* ***role model*** *whom I can learn from; I will select the good qualities and follow them."*
>
> — *Confucius, "The Analects"*

To address this question, we draw inspiration from Confucius' dictum in *The Analects*, and propose the Role-Model

Xiangyu Chen, Chuhao Zhou, Yuxi Liu, and Jianfei Yang are with the Multimodal AI and Robotic Systems (MARS) Lab, School of Mechanical and Aerospace Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798. Email: {xiangyu014, chuhao002, yuxi002}@e.ntu.edu.sg; jianfei.yang@ntu.edu.sg.

[†]Corresponding Author.

Reinforcement Learning (RM-RL) framework. RM-RL integrates online exploration with offline supervised learning by periodically selecting a role-model action, which is the highest-reward action observed under similar initial states, and using it to guide the remaining peer actions. The peer actions are automatically labeled by reference to the role-model action, allowing the online samples to be reformulated into supervised training data. These labeled samples are then repeatedly reused in offline training, ensuring that valuable experiences contribute to policy improvement multiple times rather than only once. This selection-and-labeling process is performed periodically during online interaction, which enables the framework to continuously inject stable supervised signals into the training loop. As a result, RM-RL substantially enhances data efficiency and mitigates distribution mismatch [17], [18], ultimately achieving both stable optimization and robust policy learning for high-precision robotic manipulation in the real world.

To validate the effectiveness of the RM-RL framework in real-world applications, we consider a precise manipulation task: placing a cell plate into a designated slot with millimeter-level accuracy, such as a cell plate shelf, as shown in Fig. 1. At each iteration, the policy network predicts actions based on the real-world environmental observation, which are then transformed to control commands for the robotic arm to execute. The reward is computed as the deviation between the executed and target poses and used to update the policy network via policy gradient. The role-model samples are simultaneously identified to label online training data, enabling their reuse during offline training. The offline training operates in a supervised manner, iteratively fine-tuning the policy network to improve performance. The contributions of this paper are summarized as follows:

- We propose a role-model mechanism that, by sampling under similar initial conditions, identifies approximately optimal actions to label real-world samples collected during online RL training. This labeling process transforms offline learning into a supervised paradigm, thereby enhancing the efficiency of real-world RL training.
- We propose a combined online–offline RL framework in which data are collected during online training and subsequently labeled through the role-model mechanism. Within this framework, each sample obtained from online training is reused multiple times in the offline training stage, thereby enhancing data efficiency.
- The experiment results demonstrate that the proposed role-model strategy and recipe can improve the sampling data efficiency, get faster convergence, and achieve a better success rate in the real-world, precise tasks.

II. Related Works

A. Reinforcement Learning

Reinforcement learning (RL) aims to train a policy to make decisions based on the reward generated from the environment, typically modeled as a Markov decision process (MDP) [19]. Over the past decade, RL has demonstrated its power in combination with deep neural networks in multi-agent systems [20] and robotic areas, such as autonomous driving and robot locomotion tasks. However, RL still faces challenges when using real-world data for training [21].

Traditional RL methods can be broadly categorized as online RL, where the policy model is updated simultaneously with the environment's reward during the training process. Due to the emergence of high-performance simulators, like Isaaclab [10], Mujoco [9], online RL has demonstrated strong power in robotic locomotion, with Q value-based approaches (e.g., Q-learning and DQN [12]), policy gradient methods [22], and actor-critic frameworks (e.g., A3C [23], SAC [24]). Despite their success, online approaches often suffer from high sample complexity and safety concerns, making their deployment in real-world systems such as robotics and autonomous driving particularly challenging.

To address the limitations of costly and risky online interaction, offline RL (also referred to as batch RL) has gained increasing attention. In offline RL, policies are trained entirely from previously collected datasets without additional environment access, allowing the reuse of experiences from demonstrations, simulators, or operational logs. Recent advances, including CQL [25], and IQL [26], have made notable progress toward improving stability and mitigating the risk of extrapolation errors, showing promising results in safety-critical domains such as healthcare, recommendation, and robot manipulation. Specifically, Zhou et al. [27] leverage an extensive dataset to improve the generalization capabilities. Nevertheless, offline RL still faces significant challenges in practice. Widely used offline RL datasets [28]–[30] are almost collected in simulation environments, where there exist significant gaps between the real-world settings. Collecting large-scale and high-quality real-world robotic datasets [31] remains difficult and resource-intensive, while the issue of distributional shift persists, as learned policies may generate out-of-distribution actions unsupported by the dataset. These limitations continue to hinder the scalability and reliability of offline RL in real-world applications.

B. Reinforcement Learning for Real-world Robotic Tasks

Reinforcement Learning (RL) has demonstrated remarkable capabilities in robotic domains such as locomotion, manipulation, and autonomous navigation. Quadruped and humanoid robots have successfully employed deep RL to achieve agile locomotion directly on hardware [32], [33], while mobile robots have leveraged RL for navigation in unstructured environments [34], [35]. In manipulation, RL has enabled large-scale robotic grasping [36]–[38], underscoring its potential for practical deployment.

Despite these successes, deploying RL in the real world remains challenging [21] due to issues of sample inefficiency, safety, and robustness to domain shifts. Collecting large amounts of on-robot data is expensive and risky, and simulators often fail to capture fine-grained physical properties, resulting in significant sim-to-real gaps [39], [40]. To address these challenges, researchers have developed approaches such as sim-to-real transfer with domain randomization [41], model-based RL with latent world models for efficient data usage [42], and offline RL leveraging prior robot datasets for safe policy training [26]. Moreover, hybrid frameworks that combine demonstrations with online fine-tuning [43] or integrate classical controllers with RL [44] further reduce the interaction burden and improve training stability.

III. Methodology

A. Problem Definition

As we mentioned in Sec. I, our work addresses the robotic manipulating task requiring millimeter-level accuracy. The task involves using the robotic arm's bio-gripper to pick and place a cell plate into a designated position of comparable size. We formalize the task as a standard one-step Markov Decision Process (MDP) [19], $\mathcal{M} = \big(\mathcal{S}, \mathcal{A}, p, \mathcal{R}, \rho_0, \gamma\big)$. Specifically, the state $\mathcal{S}$ includes the environmental images I and the corresponding estimated picking poses P^e of the robotic arm. The action $\mathcal{A}$ is the set of all candidate pose adjustments ΔP to P^e. For clarity, we denote each pose adjustment ΔP as an action $a \in \mathcal{A}$. The reward function $\mathcal{R}$ provides the real-world reward as the deviation between the final and target poses. Since the selected task is to pick and place the cell plate on a planar surface, we only consider the pose components x, y, and ψ, denoted as $P = [x, y, \psi]$. Here, x and y represent translations along the x- and y-axes, and ψ denotes the rotation about the z-axis. Thus, the action can be simplified to $a = [\Delta x, \Delta y, \Delta \psi]$.

B. Overall Framework

Fig. 2 illustrates the proposed framework of the proposed Role-Model Reinforcement Learning (RM-RL). The framework adopts the hybrid online-offline training paradigm and consists of three main components: Online Real-World RL Training, Role-Model Online Labeling, and Data Replay. Before training begins, the cell plate is placed at the target area and its pose is recorded as the target pose P^{target}, which remains fixed throughout the process. During Online Real-World RL Training, actions are sampled under similar initial states in scene i and used to update the policy network π_θ via policy gradient. At step t, the global camera captures a color image I_t, from which the estimated cell plate pose $P^e_{i,t}$ is obtained. The pair $(I_{i,t}, P^e_{i,t})$ is then fed into the policy network to predict an adjustment pose $\Delta P_{i,t}$, which corresponds to the action $a_{i,t}$ defined above. The robotic arm executes the final picking pose, which is defined as:

$$P^{pick}_{i,t} = P^e_{i,t} + a_{i,t}. \tag{1}$$

Once the plate is successfully grasped, the robotic arm follows a pre-planned trajectory to place it at the center of the target area. The resulting pose $P^{final}_{i,t}$ is recorded, and the reward is computed by the reward function $\mathcal{R}$, based on the deviation between P^{final}_t and P^{target}. This reward is used to update the policy network with the policy gradient algorithm [22]. In the Role-Model Online Labeling, we leverage the proposed role-model strategy to online label the training data from scene i as $\mathcal{D}_i$, incrementally building a well-labeled dataset $\mathcal{D}$ for further offline RL training. The offline training processes are executed multiple times in the Role-Model Online Labeling and Data Replay. The details of the core contributions, role-model online labeling and online-offline training recipe, are presented in the following section.

C. Role-Model Reinforcement Learning

This part illustrates the Role-Model Reinforcement Learning (RM-RL) in detail, including the role-model online labeling and the hybrid online-offline RL training. Role-model labeling enables self-annotation to improve training efficiency, while hybrid online–offline RL training emphasizes reusing scarce training data to enhance data efficiency.

1) Role-Model Online Labeling: The role-model online labeling selects the role-model action to label the corresponding samples within each scene in the online training stage. During the step-by-step online training, we group the steps with similar initial states $\mathcal{S}_i$, their rewards $\mathcal{R}_i$, and actions $\mathcal{A}_i$ to formulate the scene i. The role-model strategy is to select the action with the best reward as the role-model action a^*_i from $\mathcal{A}_i$, which can be formulated as follows:

$$a^*_i = \arg\max_{a_{i,k} \in \mathcal{A}_i} \mathcal{R}(a_{i,k}, s_{i,k}), \tag{2}$$

where $a_{i,k}$ and $s_{i,k}$ are the action and state in scene i at step k. Then, we can use the role-model action a^*_i as the approximately optimal action to label the remaining states in scene i. Since the pose adjustment prediction is formulated as a discrete probability distribution, the variables Δx, Δy, and $\Delta \psi$ are restricted to a fixed set of candidate values. Consequently, the selection of the pose adjustments can be transformed into a multi-class classification problem, where predictions of $\Delta x, \Delta y$, and $\Delta \psi$ can be mapped into corresponding discrete classes, respectively. The indices of the $\Delta x, \Delta y$ and $\Delta \psi$ of the role-model action a^*_i are extracted as $\mathcal{I}_i = [ind_x, ind_y, ind_\psi]$. Then, the states are collected in scene i and labeled with $\mathcal{I}_i$ to incrementally build the dataset $\mathcal{D}$. The dataset collection process can be represented as:

$$\mathcal{D}_i = \{(s_{i,k}), \mathcal{I}_i\}^N_{k=1}, \tag{3}$$

$$\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_i\}, \tag{4}$$

where $\mathcal{D}_i$ and $\mathcal{I}_i$ represent the dataset and the index label for the i-th scene with similar initial states, N represents the total number of the state and s^k_i is one state in i-th scene. The well-labeled dataset $\mathcal{D}$ can be regarded as demonstrations for executing supervised learning to update the policy network.

2) Hybrid Online-Offline RL Training: To enhance training efficiency, we adopt a hybrid online–offline framework that facilitates the effective reuse of sampled data. In online training, the robotic arm interacts with the real-world environment in real-time to explore and update the policy network π_θ, while the offline training further fine-tunes the policy using role-model–labeled data. We elaborate the details about online training, offline training, network architecture, and reward design in the following.

Online Training: The online RL adopts the policy gradient [45] fashion to train the policy network, since other

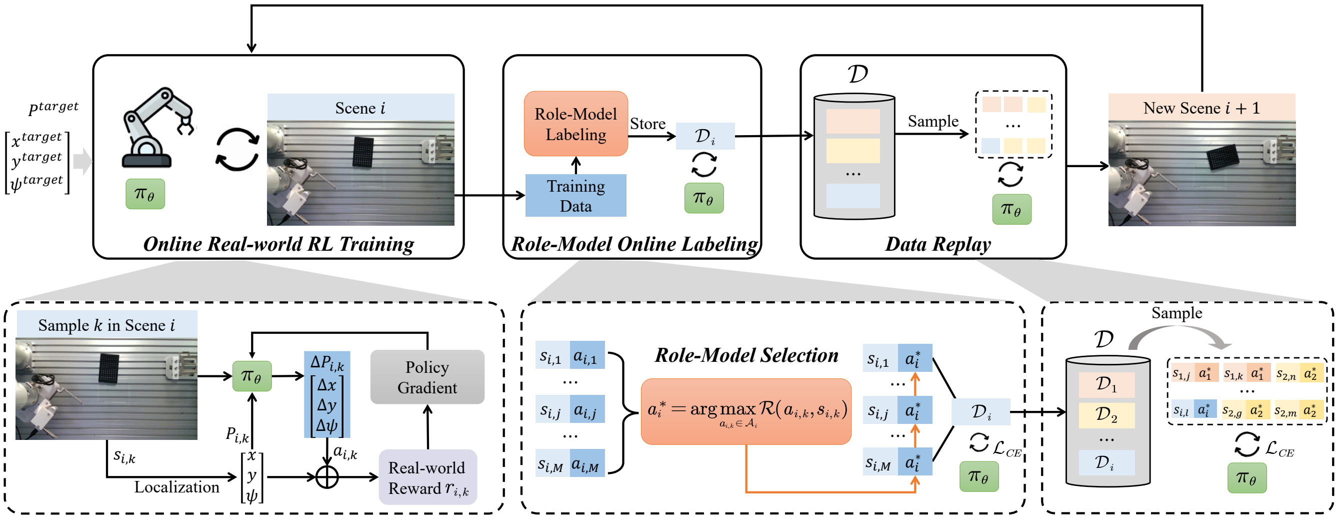

Fig. 2: The framework of the proposed Role-Model Reinforcement Learning (RM-RL). The framework mainly includes three parts: the Online Real-world RL training, the Role-Model Online Labeling, and the Data Replay. In the online real-world RL part, actions are sampled to update the policy network using a policy gradient under similar scenes and initial states. In the role-model online labeling part, the approximately optimal action a_i^* is selected as a label for the rest of the states. The labeled data are collected as the sub-dataset $\mathcal{D}_i$ to build a dataset $\mathcal{D}$ for further training. In the data replay part, actions and states are sampled from the $\mathcal{D}$ to update the policy network.

training methods, like Actor-Critic [46], [47], Proximal Policy Optimization [48], require extensive action samples to achieve convergence, which is time-consuming in the real world. Online RL aims to maximize the following objective:

$$J(\theta) = \mathbb{E}_{a \sim \pi_\theta}\Big[R(a)\Big], \tag{5}$$

where a is the sampled action given policy π_θ and $R(a)$ is the one-step reward of action a. In policy gradient, we instead minimize the surrogate loss function for training:

$$\mathcal{L}(\theta) = -\mathbb{E}_{s,a}\Big[\log \pi_\theta(a \mid s)\, R(a)\Big]. \tag{6}$$

Offline Training: To enhance the data efficiency and generalization capability of the policy, we separately perform offline RL training in the Role-Model Online Labeling and Data Replay. Both offline RL training adopt the same loss function, while the distinction lies in the datasets. The first training is conducted on $\mathcal{D}_i$, and the second on $\mathcal{D}$ periodically. After collecting $\mathcal{D}_i$, the first offline training is applied, allowing the policy to learn the approximately optimal solution in the i-th scene. We then perform the second offline training at fixed step intervals, ensuring that the policy periodically replays the information from the whole dataset $\mathcal{D}$. The training data are randomly sampled from the $\mathcal{D}$ during the training process.

Specifically, given a state s from scene j and its indices $\mathcal{I}_j$ in $\mathcal{D}$, the corresponding optimal action is $a^* = \mathcal{A}_j(ind_x, ind_y, ind_\psi)$. Consequently, the action distribution specializes to a deterministic form for x, y, and ψ dimensions. For the i-th action dimension, the probability of the $a^*[i]$ is 1, while the probabilities of other candidate actions are set to 0. This yields a one-hot distribution in the action space for each dimension, expressed as:

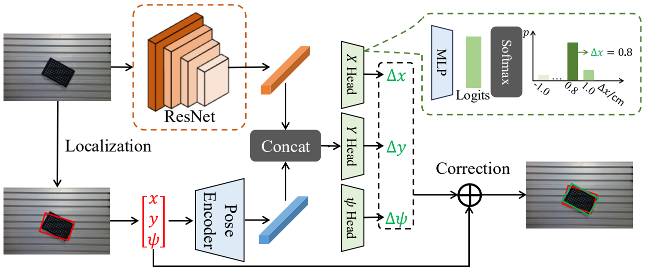

Fig. 3: The Policy network. The network takes the global image and the estimated picking pose as inputs, and outputs probability distributions over Δx, Δy, and $\Delta \psi$. The final pose adjustments are sampled from these distributions to refine the picking pose.

$$\pi^*(a[i] \mid s) = \begin{cases} 1, & \text{if } a[i] = a^*[i], \\ 0, & \text{if } a[i] \neq a^*[i]. \end{cases} \quad i = 0, 1, 2. \tag{7}$$

Then, the cross-entropy loss can be calculated as:

$$\mathcal{L}_\theta = -\sum_{i=0}^{2} \log \pi_\theta(a^*[i] \mid s). \tag{8}$$

Additionally, this offline training scheme can be applied during a pre-training stage, where the policy network is updated using data collected from prior real-world experiments. The resulting pretrained policy offers a favorable initialization for subsequent reinforcement learning, leading to faster convergence and improved training stability.

Network Design: The architecture of our policy network π_θ is shown in Fig. 3. There are two inputs of the policy network, the color image I and the estimated pose P^e. The output of the policy network is the adjustment pose ΔP, referred to as the action $a \in \mathcal{A}$ introduced earlier, adjusting P^e for precise picking. ResNet [49] is leveraged to extract the image feature, and an MLP is used to encode the input pose. Then, the image and pose features are concatenated and fed into three separate output heads, which respectively predict the translation adjustments along the x- and y-axes, and the rotation adjustment around the z-axis. All the heads adopt the same MLP architecture to predict logits, which are transformed into probability distributions using the softmax function. The adjustments Δx, Δy, and $\Delta\psi$, sampled from the predicted distributions, are represented collectively as the action $a = [\Delta x, \Delta y, \Delta\psi]$. Finally, the final picking pose is calculated as $P^{pick} = P^e + a$, which is executed by the robotic arm to pick the cell plate precisely.

Reward Design: The reward is designed to evaluate the similarity between the target pose P^{target} and the final pose P^{final} after executing the actions. The similarity measure separately accounts for translation and rotation errors. For translation, the error e_{trans} is defined as the Euclidean distance between the translation in the final pose t^{final} and the target pose t^{target}, which can be formulated as follows:

$$e_{trans} = \|t^{final} - t^{target}\|_2. \tag{9}$$

For rotation, since the task only requires planar alignment, we only consider the yaw angle in the final and target pose for error calculation. The cosine value of the angle difference is used to measure the orientation similarity, and the corresponding rotation error can be formulated as:

$$e_{rot} = 1 - \cos(\psi^{final} - \psi^{target}). \tag{10}$$

The overall RL reward combines the translation and rotation error, which can be calculated as:

$$r = \exp(-(e_{trans} + e_{rot})). \tag{11}$$

The exponential operation is applied to normalize the final reward within the range $[0, 1]$, where the reward increases as the combined translation and rotation errors decrease. In the real-world experiment, we first put the cell plate in the target position and use the overhead camera to localize it, getting the t^{target} and ψ^{target}. After the robotic arm finishes executing actions, we re-localize the cell plate to get the t^{final} and ψ^{final} and calculate the reward.

In summary, the Role-Model Reinforcement Learning (RM-RL) framework integrates online and offline training through a role-model strategy that provides labeled data for offline training. Unlike traditional RL with replay buffers [12], which suffers from distribution mismatch [11], [15], RM-RL constrains the policy network π_θ to move forward to the approximately optimal action among similar initial states by supervised learning, significantly improving sampling efficiency, adaptability, and stability.

D. Cell Plate Localization

To get the estimated pose P^e of the cell plate from the real-world observation, a general pipeline consisting of object detection, segmentation, and pose estimation is leveraged. For precisely localizing the cell plate, we use a global camera instead of some physical labels, providing both the environmental color image I_{RGB} and depth image I_{depth}. The Grounding Dino [50], a pretrained open-vocabulary object detection method, is then utilized to localize the bounding box of the plate $(x_{min}, y_{min}, x_{max}, y_{max})$ via the prompt "black cell plate". Furthermore, we adopt Segment Anything Model (SAM) [51], a foundation model of object segmentation, to predict the mask area $\mathcal{M}$ from the bounding box of the target cell plate. The central pixel coordinate is denoted as (u, v), where $u = \frac{x_{min}+x_{max}}{2}$ and $v = \frac{y_{min}+y_{max}}{2}$. Then, the depth value is the average depth within the cell plate mask, which can be calculated as:

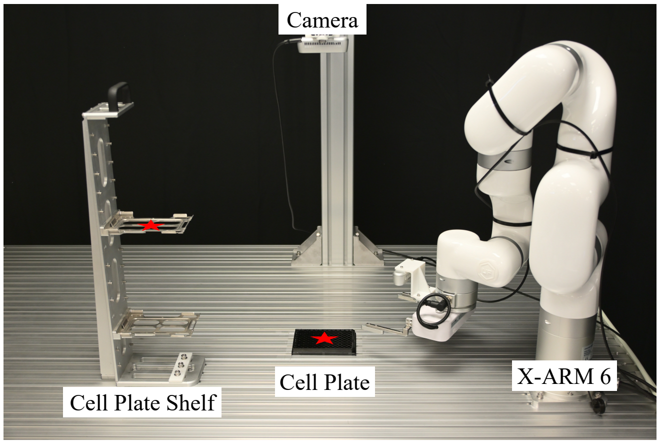

Fig. 4: The real-world experimental setting. The setting includes robotic arm (X-ARM 6), cell plate, cell plate shelf, and overhead camera (Intel RealSense D435). The red stars represent the target positions for the precise picking and placing tasks.

$$Z = mean(I_{depth} \odot \mathcal{M}), \tag{12}$$

where $\odot$ represents the Hadamard product. Then, the 3D position $\mathbf{p}_{camera} = [X, Y, Z]$ of the cell plate in the camera coordinate system can be formulated as:

$$X = \frac{(u - c_x) \cdot Z}{f_x}, Y = \frac{(u - c_y) \cdot Z}{f_y}, Z = Z, \tag{13}$$

where (f_x, f_y) are the focal lengths in pixel units and (c_x, c_y) is the principal point. After recovering the $\mathbf{p}_{camera}$ in the camera coordinate system, its position in the world coordinate system is obtained using the camera extrinsic parameters. With rotation matrix R and translation vector t, its position $\mathbf{p}_{world}$ in the world coordinate system is:

$$\mathbf{p}_{world} = R \cdot \mathbf{p}_{camera} + t. \tag{14}$$

Then, the yaw angle ψ can be estimated from $\mathcal{M}$ by applying Principal Component Analysis (PCA) [52], where the first principal component indicates the major directions. Finally, we use the combination of $\mathbf{p}_{world}$ and ψ as the pick pose of the end-effector in the robotic arm to finish the picking task.

IV. Experiments

The experiment consists of two main parts. First, we compare the training performance of our framework with other RL methods, reflecting the effectiveness of the proposed role-model mechanism in improving data efficiency and training efficiency. Second, we evaluate the performance of the selected methods in two real-world tasks, requiring precise picking and placing. These real-world experiments further demonstrate that the proposed method can be effectively applied to practical robotic applications.

A. Hardware

For the hardware, as illustrated in Fig. 4, we use a 6-DOF Ufactory X-ARM 6 with a bio gripper as our platform to finish the designed experiments. RealSense D435 is mounted overhead to provide depth and RGB visual perception, enabling the system to detect and localize the cell plate in the workspace. Two target positions, marked as red stars, are defined in the setup: one is located on the table surface (where the cell plate is placed), and the other is on the shelf slot. This configuration enables evaluation of reinforcement learning–based manipulation policies under real-world conditions, requiring accurate transfers between different spatial levels. For the calibration of the overhead RealSense camera, we first calibrate the camera on the robotic wrist through chessboard calibration method [53] to get the transformation matrix T_{world}^{wrist}. Then, the wrist camera and the overhead camera localize the chessboard simultaneously, to get the transformation matrix T_{wrist}^{camera}. Finally, we get the transformation matrix $T_{world}^{camera} = T_{wrist}^{camera} \cdot T_{world}^{wrist}$.

B. Baseline

To verify the effectiveness of the proposed method, we compare the performance of the standard RL [22], standard RL with replay buffer (RL + Replay Buffer) [12], the proposed Role-Model Reinforcement Learning (RM-RL), and pretrained RM-RL. To ensure fair comparisons, all RL methods use the same policy network and are trained with the same policy gradient-based method and training configurations. Since the selected task is relatively simple, we adopt policy gradient for optimization instead of more advanced algorithms such as PPO or Actor-Critic. It is worth noting that our proposed role-model strategy is orthogonal to these algorithms and can be readily integrated into them in future work. For the pretrained RL + MR, we collect about 200 samples, which are labeled through the proposed role-model strategy, and the policy model is trained before the beginning of real-world RL training. To ensure consistency, all policy networks were trained and evaluated on the same PC, equipped with an NVIDIA RTX 4060 GPU (6 GB).

C. Training Performance

We record the reward changes of the selected methods during the RL training, and apply the Exponential Moving Average algorithm to smooth the reward sequences, providing a clearer visualization of the performance trend. Since each trial takes about 2-3 minutes to finish, one successful training may take 7 hours. The results are shown in Fig. 5. As we can see from the picture, the rewards of RM-RL and Pretrained RM-RL increase quickly at the beginning and finally converge to higher reward values than the Standard RL and RL + Replay Buffer methods, demonstrating the effectiveness of the proposed role-model labeling mechanism in improving training efficiency. Compared with the reward curves of other RL methods without pretraining, the reward curve of the pretrained RM-RL shows a more consistent and stable improvement, demonstrating that reusing the previous role-model-labeled training data as a pretraining stage is effective to improve the training stability and efficiency.

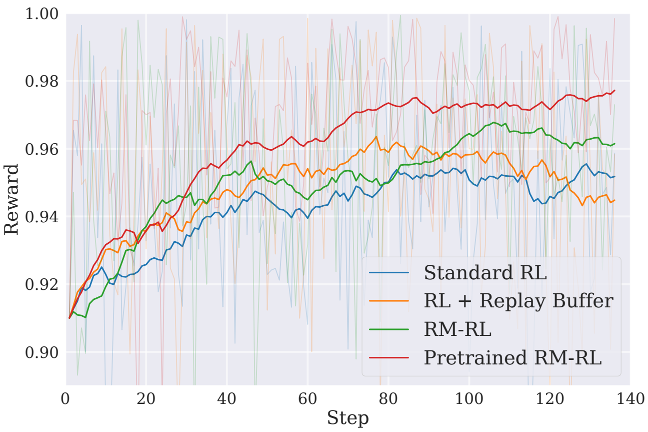

Fig. 5: The results of reward curves. The reward curves record the reward changes during real-world RL training from the selected baselines, Standard RL, standard RL with replay buffer (RL + Replay Buffer), the proposed Role-Model Reinforcement Learning (RM-RL), and pretrained RM-RL.

Methods	T_{τ}^{10}	Average Reward
Standard RL [22]	45	0.948 (0.029)
RL + Replay Buffer [12]	36	0.953 (0.031)
RM-RL	35	0.956 (0.027)
Pretrained RM-RL	**30**	**0.969** (0.023)

TABLE I: Quantitative experimental results of different RL methods evaluated by T_{τ}^{10} and average reward.

For quantitative analysis, we use two metrics to evaluate the training performance. Similar to the metric $J^{eff.}$ [21], we record the training step index at which the learned policy achieves a predefined reward threshold for the tenth occurrence. The calculation of the metric is shown as follows:

$$T_{\tau}^{10} = \min\left\{ t \in \mathbb{N} \,\middle|\, \sum_{i=1}^{t} \mathbf{1}_{\{r_i > \tau\}} = 10 \right\}, \qquad (15)$$

where τ is a predefined reward threshold, r_i is the reward in step i, and $\mathbf{1}_{\{r_i > \tau\}}$ is the indicator function, equal to 1 if $r_i > \tau$ and 0 otherwise. This metric reflects not only whether the policy can eventually reach the desired reward level, but also how rapidly and consistently it does so across training. By requiring the threshold to be reached multiple times, the metric mitigates the influence of outlier fluctuations and provides a more robust measure of data efficiency and training stability. The second evaluation metric is the average reward, defined as the mean of the rewards obtained over the training steps. This metric reflects the overall training performance of the selected training methods.

Table I summarizes the quantitative experimental results. In particular, Pretrained RM-RL achieves the best results, requiring only 30 steps to reach the threshold for the 10th time (a 33% reduction from Standard RL) and attaining the highest average reward of 0.969 with the lowest variance,

Methods	Average Reward	e_{trans}/mm	e_{rot}/°
Standard RL	0.964 (0.024)	3.6 (0.27)	**0.58** (0.49)
RL + Replay Buffer	0.944 (0.030)	5.8 (0.33)	1.01 (0.43)
RM-RL	0.970 (0.025)	3.0 (0.30)	0.89 (0.38)
Pretrained RM-RL	**0.978** (0.012)	**2.1** (0.12)	0.60 (0.45)

TABLE II: Performance comparison of different reinforcement learning strategies in real-world robotic picking and placing tasks, evaluated by average reward, position error, and rotation error.

indicating faster convergence and more stable learning. Although the RL with Relay Buffer demonstrates competitive performance, it exhibits the largest standard deviation in experimental results, indicating instability in training.

D. Real-World Performance

To evaluate the pick and place performance in the real-world environment, we conduct two experiments. The first experiment follows the training process, putting one cell plate into the given box of a similar size. We provide 10 random initial positions of the cell plate. The robot needs to pick the cell plate from the initial position and place it in the target position. We record the average reward, translation error e_{trans}, and rotation error e_{rot} in Table II of the compared methods for evaluation. From the experimental results, the Pretrained RM-RL achieves the highest average reward (0.978). Its translational error is reduced to 2.1 mm, representing a 42% improvement over Standard RL (3.6 mm) and a 64% reduction compared with RL + Replay Buffer (5.8 mm). For rotational accuracy, Pretrained RM-RL (0.60°) is nearly identical to Standard RL (0.58°) and 41% better than RL + Replay (1.01°). Overall, these results demonstrate that pretraining using role-model-labeled data significantly enhances real-world pick-and-place performance.

The other experiment is placing a cell plate onto the cell plate shelf, given random initial states. The aim of this experiment is to evaluate whether the predicted pose adjustment is also effective for other tasks that also need precise operations, and the influence of the small errors on these RL-based methods. The robotic arm autonomously picks the cell plate with the localization information and the adjustment pose from RL. A fixed trajectory is designed to guarantee the cell plate can be put on the cell plate shelf with a proper grasping pose. We evaluate the performance of different RL-based methods by measuring the success rate over ten trials. The experiment results are summarized in Table III. From the experimental results, the Standard RL achieves a success rate of 50%, while RL + Replay performs worse with only 40%, demonstrating the replay with distribution transit error may fail the precise tasks. RM-RL also shows unsatisfactory results, achieving only a 70% success rate, despite presenting results comparable to Pretrained RL + MR in the previous experiment. The Pretrained RM-RL succeeds in all trials, demonstrating the effectiveness of the proposed receipt with pretraining in enhancing the robot's capability for precise tasks.

Methods	Successful Rate
Standard RL	50%
RL + Replay Buffer	40%
RM-RL (Ours)	70%
Pretrained RM-RL (Ours)	**100%**

TABLE III: Success rates of different reinforcement learning strategies in real-world picking a cell plate on a shelf.

V. Summary and Future Work

This paper addresses the challenge of achieving precise robot manipulation in real-world applications, requiring millimeter-level accuracy. Online reinforcement learning (RL) can improve safety and accuracy but requires extensive sampling in real-world environments, which is often impractical. Offline RL reduces the need for real-world data collection but is hindered by the difficulty of acquiring high-quality datasets and the distribution shift between offline data and real-world execution. To overcome these limitations, we proposed Role-Model Reinforcement Learning (RM-RL), a framework that integrates online and offline training. The role-model strategy labels online samples with approximately optimal actions, creating reliable datasets for offline training and policy pretraining. This approach enhances both data and training efficiency. Experimental results demonstrate that RM-RL achieves faster convergence, more stable training, and notable accuracy improvements (53% in translation and 20% in rotation) while consistently accomplishing a challenging task: placing a cell plate onto a shelf, which competing RL algorithms fail to perform reliably.

In future work, we will extend RM-RL to more complex and long-horizon robotic tasks that demand sequential decision-making, generalization capability, and sustained precision across multiple stages. Such scenarios will provide a rigorous benchmark to evaluate the scalability of our framework and further establish its potential as a robust, data-efficient solution for real-world robotic applications.

VI. Acknowledgments

This work is supported by a Start-up Grant from Nanyang Technological University and jointly funded by the Singapore Ministry of Education (MOE) under a Tier-1 research grant.

References

[1] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *Conference on Robot Learning*. PMLR, 2023, pp. 2165–2183.

[2] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, *et al.*, "π_0 : A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[3] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[4] Z. Lan, Y. Jiang, R. Wang, X. Xie, R. Zhang, Y. Zhu, P. Li, T. Yang, T. Chen, H. Gao, *et al.*, "Autobio: A simulation and benchmark for robotic automation in digital biology laboratory," *arXiv preprint arXiv:2505.14030*, 2025.

[5] S. Li, Y. Huang, C. Guo, T. Wu, J. Zhang, L. Zhang, and W. Ding, "Chemistry3d: Robotic interaction benchmark for chemistry experiments," *arXiv preprint arXiv:2406.08160*, 2024.

[6] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[7] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, no. 1, pp. 297–330, 2020.

[8] H. Geng, F. Wang, S. Wei, Y. Li, B. Wang, B. An, C. T. Cheng, H. Lou, P. Li, Y.-J. Wang, *et al.*, "Roboverse: Towards a unified platform, dataset and benchmark for scalable and generalizable robot learning," *arXiv preprint arXiv:2504.18904*, 2025.

[9] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[10] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

[11] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[14] M. Rowland, R. Dadashi, S. Kumar, R. Munos, M. G. Bellemare, and W. Dabney, "Statistics and samples in distributional reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5528–5536.

[15] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International conference on machine learning*. PMLR, 2019, pp. 2052–2062.

[16] Y. Liu, A. Swaminathan, A. Agarwal, and E. Brunskill, "Provably good batch off-policy reinforcement learning without great exploration," *Advances in neural information processing systems*, vol. 33, pp. 1264–1274, 2020.

[17] A. Kumar, J. Hong, A. Singh, and S. Levine, "Should i run offline reinforcement learning or behavioral cloning?" in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=AP1MKT37rJ

[18] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN computer science*, vol. 2, no. 3, p. 160, 2021.

[19] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[20] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.

[21] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv preprint arXiv:1904.12901*, 2019.

[22] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[23] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PmLR, 2016, pp. 1928–1937.

[24] T. Degris, P. M. Pilarski, and R. S. Sutton, "Model-free reinforcement learning with continuous action in practice," in *2012 American control conference (ACC)*. IEEE, 2012, pp. 2177–2182.

[25] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative q-learning for offline reinforcement learning," *Advances in neural information processing systems*, vol. 33, pp. 1179–1191, 2020.

[26] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv preprint arXiv:2110.06169*, 2021.

[27] G. Zhou, L. Ke, S. Srinivasa, A. Gupta, A. Rajeswaran, and V. Kumar, "Real world offline reinforcement learning with realistic data source," *arXiv preprint arXiv:2210.06479*, 2022.

[28] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.

[29] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4rl: Datasets for deep data-driven reinforcement learning," *arXiv preprint arXiv:2004.07219*, 2020.

[30] R. Rafailov, K. Hatch, A. Singh, L. Smith, A. Kumar, I. Kostrikov, P. Hansen-Estruch, V. Kolev, P. Ball, J. Wu, *et al.*, "D5rl: Diverse datasets for data-driven deep reinforcement learning," *arXiv preprint arXiv:2408.08441*, 2024.

[31] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, *et al.*, "Bridgedata v2: A dataset for robot learning at scale," in *Conference on Robot Learning*. PMLR, 2023, pp. 1723–1736.

[32] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[33] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," *Science Robotics*, vol. 9, no. 89, p. eadi9579, 2024.

[34] L. Wijayathunga, A. Rassau, and D. Chai, "Challenges and solutions for autonomous ground robot scene understanding and navigation in unstructured outdoor environments: A review," *Applied Sciences*, vol. 13, no. 17, p. 9877, 2023.

[35] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot navigation in crowded environments using deep reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5671–5677.

[36] Y. Xu, W. Wan, J. Zhang, H. Liu, Z. Shan, H. Shen, R. Wang, H. Geng, Y. Weng, J. Chen, *et al.*, "Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4737–4746.

[37] W. Wan, H. Geng, Y. Liu, Z. Shan, Y. Yang, L. Yi, and H. Wang, "Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 3891–3902.

[38] Y. Geng, B. An, H. Geng, Y. Chen, Y. Yang, and H. Dong, "Rlafford: End-to-end affordance learning for robotic manipulation," in *2023 IEEE International conference on robotics and automation (ICRA)*. IEEE, 2023, pp. 5880–5886.

[39] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2020, pp. 737–744.

[40] F. Muratore, F. Ramos, G. Turk, W. Yu, M. Gienger, and J. Peters, "Robot learning from randomized simulations: A review," *Frontiers in Robotics and AI*, vol. 9, p. 799893, 2022.

[41] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[42] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *arXiv preprint arXiv:1912.01603*, 2019.

[43] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[44] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6023–6029.

[45] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[46] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[47] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[48] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[50] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, J. Yang, Q. Jiang, C. Li, J. Yang, H. Su, *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," in *European conference on computer vision*. Springer, 2024, pp. 38–55.

[51] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," *arXiv:2304.02643*, 2023.

[52] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[53] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2002.