2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Multi-Level Progressive Reinforcement Learning for Control Policy in Physical Simulations

Kefei Wu[1], Xuming He[1], Yang Wang[1] and Xiaopei Liu[1,*]

***Abstract*— Training model-free intelligent agents in complex real-world scenarios using reinforcement learning (RL) often necessitates simulation-based environments due to high physical expenses. However, when simulation takes a long time, e.g., in an unsteady 3D fluid simulation with interactions to the controllable solids, existing RL algorithms meet difficulty to accomplish training within a reasonable timeframes. In this paper, we propose a novel multi-level framework for RL to accelerate convergence as the first attempt to address this difficulty. Motivated by the idea of multi-grid solver, the control policy on a virtual agent over time can be decomposed into different frequency levels, which can be progressively learned via a set of simulations in a coarse-to-fine manner. It is expected that most RL trials are performed in coarser simulations to learn lower control frequency levels with more efficient convergence, while higher frequency levels require much less RL trials, thus significantly accelerating the learning process. To implement our idea, we designed a novel multi-level residual network with a filter module attached, where each level of the network is learned by performing RL for a given simulation resolution. The proposed framework is evaluated by conducting policy learning experiments on a virtual aerial (2D) and an underwater (3D) robot, both requiring time-consuming physical simulations. Our results demonstrate a decrease in almost half in learning time compared to a direct RL approach, while achieving similar control performance.**

I. INTRODUCTION

Physical simulations have already been an indispensable tool for studying complex robot control policies, especially within the reinforcement learning (RL) framework, mainly due to high physical expense in reality. Although rigid body dynamics has been frequently employed in many platforms, such as PyBullet [1], other simulations required for training, such as deformable body [2] or fluid [3] simulations, are less common due to high computational cost involved. Even for rigid body dynamics, if precision is required, e.g., in a virtual-to-real learning task where contact forces must be accurately solved, the simulation could still be prohibitively slow [4], [5]. Consequently, trade-offs are often made between accuracy and efficiency in simulations for robot training, limiting the ability to explore learning algorithms that can ultimately be applied to real agents. Although some new simulation methods have emerged, e.g., a GPU-optimized lattice Boltzmann solver for fluid-solid coupling that can be much faster than a traditional counterpart [6], [7], simulations commonly used are still not yet ready for extensively applying existing RL algorithms, especially in complex and costly-to-simulate environments. Simulations in such environments can take several days or even weeks for a single learning task [8], rendering control policy learning impractical in such virtual environments.

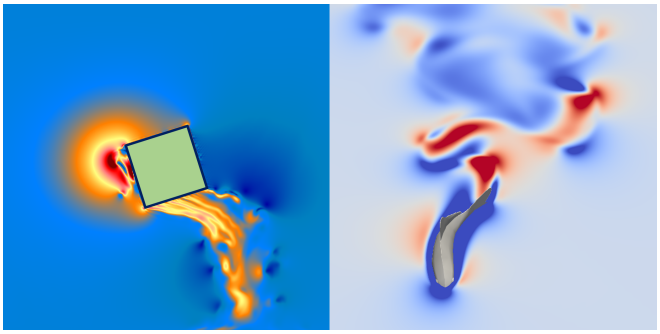

Fig. 1. **Example tests to validate our progressive RL framework.** The figures above show the instantaneous illustrations of the controlled robots in two simulated fluid-solid coupled environments which are computationally costly — a 2D jet-controlled robot in air (left) and a 3D swimming fish in water (right), where the fluid velocity field are shown by colormaps, with red color indicating high velocity and blue color for low velocity. With our method, desired control policies can be effectively learned, with training time reduced by 46.2% and 57.1% respectively for these two tests compared with a direct RL approach.

In this paper, we aim to propose the first attempt to address the challenge of prolonged learning time for simulations that entail high computational costs. Our approach is inspired by a general insight that many virtual simulations can be conducted in different resolutions, i.e., the number of samples in space and time. While high-resolution simulations are typically slow in practice, their efficiency can be considerably enhanced through a series of low-resolution simulations during the training process, which is similar as in a multi-grid solver [9], which seeks a multi-level convergence to speed up. We hypothesize that in any given simulation, the acquisition of a lower-frequency control policy can be expedited by learning in a lower-resolution simulation from the lower-frequency input states. Furthermore, as the simulation resolution increases, the control policy can progressively adapt itself by learning residuals based on the previously learned policies, but with a reduced number of trials. Consequently, we anticipate a noteworthy improvement in overall efficiency.

In this work, we propose a multi-level progressive RL algorithm, particularly for control policy learning in simulations with high computational cost. To achieve this goal, our method introduces a filter module, which can be pre-trained for different simulation resolutions, in order to align input state estimates with the respective simulations. We then incorporate the pre-trained filter module into a new multi-level

[1] ShanghaiTech University
[*] Corresponding author
Contact: {wukf, hexm, wangyang4, liuxp}@shanghaitech.edu.cn

policy learning network to bridge the gap between different simulation resolutions with the help of the residual policy blocks. To enhance sampling efficiency and convenience, the training of our policy network is based on a modified version of the popular Soft-Actor-Critic (SAC) algorithm [10], [11].

To validate the proposed framework, two scenarios are considered, both employing two-way fluid-solid coupling during the simulation. The first scenario involves a 2D box-like robot propelled by a jet flow for attitude control, as shown in Fig. 1 (left), where the simulation is conducted with a high Reynolds number in a high resolution. The second scenario involves a 3D fish-like robot for swimming control, as shown in Fig. 1 (right). We evaluate the effectiveness of our algorithm by comparing the performance of our learned policy as well as the learning efficiency to different approaches for RL, showing that our method reduces training time while maintaining satisfactory control performance. In summary, this paper makes the following contributions:

- A novel RL framework designed to expedite the training process in computationally costly simulations.
- An autoencoder-based filter considering feedback characteristics for simulations with different resolutions.
- A multi-level policy learning network is devised to account for disparities in control inputs between successive simulation resolutions, facilitating gradual alignment of the control policy with that of the ultimate high-resolution simulation.
- Two validation cases are shown to demonstrate the effectiveness of our new RL algorithm in attitude and locomotion control for robots that require fluid simulations with costly computations.

II. Related Work

A. Physical simulation in a virtual environment

Physical simulation has been intensively used in policy training, and OpenAI Gym [12] standardizes the API as the well-known interface between RL and simulators. In addition to Pybullet [1], AirSim [13] and ISAAC [14], there are plenty of efforts to narrow down the gap between simulation and reality. Combining data collected in reality with the model trained in virtual environment helps the narrowing process too in existing experiments [5], [15]. For missions influenced by fluids, building a high-fidelity environment based on fluid simulation is inevitable for accuracy, especially for fish-like robots [5], [16] and UAVs [17]. Min et al. [4] utilized finite-element method for control problems of soft-body animals, while Zhang et al. [5] used HyperFLOW [18] to mimic the 2D underwater environment. To enhance simulation efficiency for fluid, a GPU-optimized lattice Boltzmann solver for unsteady fluid flow has emerged [6], [7], enabling accuracy and efficiency, and making it possible for fluid simulation to be used in policy training.

B. Acceleration for reinforcement learning

The training efficiency of RL is always of great concern in practice. Fine-tuning model parameters or reusing current models is a straightforward idea for accelerating the training process, which has been studied for years [19]–[21]. Asynchronous methods are leveraged to build multiple workers for data collection and gradient update in large-scale parallel distributed learning [22], [23]. There are other algorithms focused on accelerations like IMPALA [24], Ape-X [25] and progressive learning [26], as implemented by applying parallel training threads or sharing samples or gradients to reduce the time cost from communication among different hardwares. Liang et al. [27] used GPU to speed up the training of challenging locomotion tasks in simulations. Rabault and Kuhnle [28] accelerated learning through a multi-environment approach. Ren et al. [8] introduced an approach to train in coarse environments and apply in finer ones to speed up training. Our work is inspired by aforementioned methods to build a novel training framework that better takes into account the differences in resolutions.

III. Preliminaries

We use the tuple $(\mathcal{S}, \mathcal{A}, p, r)$ to define a Markov decision process (MDP) with infinite horizon, where $\mathcal{S}$ and $\mathcal{A}$ refer to the continuous state and action spaces, respectively; p refers to the state transition probability and r refers to the reward given by the environment [29]. It is common to define the probability that the agent takes action $\boldsymbol{a}_t$ in state $\boldsymbol{s}_t$ as $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$ given the policy π. $\rho_\pi(\boldsymbol{s}_t, \boldsymbol{a}_t)$ denotes the state-action marginal of the trajectory's states distribution induced by the policy $\pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$.

Unlike standard RL which maximizes the expected sum of rewards $\sum_t \mathbb{E}_{(\boldsymbol{s},\boldsymbol{a}_t)\sim\rho_\pi}[\gamma^t \cdot r(\boldsymbol{s}, \boldsymbol{a}_t)]$, maximum entropy RL introduces entropy objective into the expected return to favor stochastic policies and encourage agents to take actions with potential benefits, but with higher uncertainty [30]:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\boldsymbol{s}_t,\boldsymbol{a}_t)\sim\rho_\pi}[\gamma^t \cdot (r(\boldsymbol{s}_t, \boldsymbol{a}_t) + \alpha\mathcal{H}(\pi(\cdot|\boldsymbol{s}_t)))], \quad (1)$$

where α controls stochasticity of the policy, determined according to specific tasks [10] or behaviors of the current policy [11]. $\mathcal{H}(\cdot)$ calculates the expected entropy of policy over $\rho_\pi(\boldsymbol{s}_t)$ [30]. Policy can be learned by minimizing the following expected KL divergence between the current policy's reaction and its Q-values' distributions, while narrowing differences bewteen two policies' action distributions. [10]:

$$J_\pi(\phi) = \mathbb{E}_{\boldsymbol{s}_t\sim\mathcal{D}}\left[D_{KL}\left(\pi_\phi(\cdot|\boldsymbol{s}_t)\middle|\middle|\frac{\exp(Q_\theta(\boldsymbol{s}_t,\cdot))}{Z_\theta(\boldsymbol{s}_t)}\right)\right], \quad (2)$$

where KL divergence measures the similarity between two distributions, and Z_θ normalizes Q-values' distributions. Since the policy is now represented by a neural network, we can reparameterize the policy as:

$$\boldsymbol{a}_t = f_\phi(\epsilon_t; \boldsymbol{s}_t), \quad (3)$$

where $f(\cdot)$ represents the policy network, and ϵ_t is an input noise vector, sampled from fixed distributions, which helps to generate action $\boldsymbol{a}_t$ from model distributions randomly.

IV. Multi-Level Policy Learning Framework

A. Motivation and our idea

Suppose that the simulation can be fast enough to allow substantial trials, then the control policy can converge within a reasonable period of time. However, numerous realistic scenarios feature inefficient simulators, such as tasks involving robot training with fluid interactions. In these cases, employing traditional RL methods directly often results in excessively long training time, rendering them impractical.

To tackle the aforementioned difficulty, we make the first attempt in this paper to improve training efficiency from the perspective of multi-resolution simulations. Inspired by the work in [9] and the observation that the computational cost can be significantly reduced when simulation resolution decreases, we design a novel multi-level RL framework that allows the agent to gain an efficient control policy progressively. Our core idea is that with different simulation resolutions, the control inputs can be decomposed into different frequency levels, where lower frequency levels are assumed to be well trained by the lower-resolution simulations, which can be fast. As simulation resolution rises, so does the computational expense; nevertheless, the number of RL trials required for convergence can be notably diminished, resulting in accelerated training. This concept inherently drives the development of a multi-level policy network spanning diverse resolutions. However, the inputs to each level of the network should be particularly processed, such that they match well with the corresponding simulations.

B. State filter and residual network

According to [31], the latent environmental information could be recovered from a time series of proprioceptive observations, leading to the introduction of an augmented states as the combination of proprioceptive resolution-relevant states $\boldsymbol{s}^p$ and resolution-irrelevant ones $\boldsymbol{s}^i$:

$$\begin{aligned} \boldsymbol{s}_t^p &= \{\boldsymbol{o}_{t-l+1}, \boldsymbol{o}_{t-l+2}, \ldots, \boldsymbol{o}_t\}, \\ \boldsymbol{s}_t &= \boldsymbol{s}_t^p \oplus \boldsymbol{s}_t^i, \end{aligned} \tag{4}$$

where l is the length of the sliding window on observation history; $\boldsymbol{o}_t$ is the proprioceptive observation at time t, and $\boldsymbol{s}_t^i$ are the remaining state components such as task goal and integral errors, which are concatenated with $\boldsymbol{s}_t^p$ to form the input state $\boldsymbol{s}_t$. Given a new higher simulation resolution, the augmented states defined above contain lower-frequency signals that are expected to be already learned by training in a lower-resolution simulation. Hence, it is supposed that acceleration of training can be achieved by reusing the trained policy from low-resolution. However, a direct downsampling of high-resolution simulation does not match well with the states from the corresponding low-resolution simulation, making it difficult to reuse trained policy from the low-frequency policy modules. Thus, an auto-encoder-based filter is introduced to align the current resolution-relevant states with the low-resolution ones.

The undercomplete autoencoder, a classical neural network structure as described in [32], aligns well with our needs. It endeavors to project inputs into a space formed by the corresponding resolution, thereby preserving the majority of signal characteristics among inputs. During the training process the filter using a higher-resolution simulation, an undercomplete autoencoder is attached to the front of a fixed policy module, which is the original policy obtained in the lower resolution simulation. The filter is trained by minimizing the actor loss function defined as follows:

$$J_A(\phi, \delta) = J_\pi(\phi, \delta) + \omega_F \cdot J_F(\delta), \tag{5}$$

$$\hat{\boldsymbol{s}}_t = F(\boldsymbol{s}_t^p|\delta) \oplus \boldsymbol{s}_t^i, \tag{6}$$

where the actor loss J_A contains a policy loss J_π and a filter loss J_F multiplied by a weighting factor ω_F. Filter function $F(\cdot|\cdot)$ with its parameters δ generates estimated observations $\hat{\boldsymbol{s}}_t$ as the input to the original policy.

Although the filter helps reutilize the policy obtained from low resolution, the gap between the estimated observations $\hat{s}_t^p$ and original states s_t^p are still not negligible. To address this issue, we further introduce a residual network, which serves as a compensation to the existing filter and policy modules. The residual network takes the difference between s_t^p and $\hat{s}_t^p$ as its input and outputs an action that can be added to the original actions Eq. (3) as follows:

$$\boldsymbol{a}_t = f_\phi(\epsilon_t; \hat{\boldsymbol{s}}_t) + f_{\phi'}(\epsilon_t'; (\boldsymbol{s}_t^p - \hat{\boldsymbol{s}}_t^p) \oplus \boldsymbol{s}_t^i), \tag{7}$$

where ϕ' are the parameters of a new residual policy network $f_{\phi'}$, and the inputs to the residual policy are the differences between the original observations $\boldsymbol{s}_t^p$ and the estimated ones $\hat{\boldsymbol{s}}_t^p$, concatenated with resolution-irrelevant information $\boldsymbol{s}_t^i$.

C. Multi-level policy network

Up to this point, all modules have been introduced to aid the agent in adapting to a new environment in a higher-resolution simulation. However, to facilitate training with multiple resolutions, the aforementioned procedure needs expansion to enable progressive learning across a sequence of resolutions. Building upon the approach proposed by Rusu et al. [26], which advocates freezing prior-trained policy networks to facilitate progressive learning, we also bound the state filter, policy module and residual network together and regard it as a new base policy module when transitioning to the next higher resolution. By sequentially incorporating respective modules as outlined above, a multi-level policy learning framework is constructed, which can achieve progressive learning as resolution increases, leading eventually to the determination of the final action based on Eq. (7) as:

$$\hat{\boldsymbol{s}}_t^{p,k} = F(\hat{\boldsymbol{s}}_t^{p,k+1}|\delta_k), \ \hat{\boldsymbol{s}}_t^{p,m} = \boldsymbol{s}_t^p \ , \tag{8}$$

$$\boldsymbol{a}_t = f_{\phi_0}(\epsilon_t; \hat{\boldsymbol{s}}_t^1) + \sum_{k=1}^{m-1} f_{\phi_k}(\epsilon_t'; (\hat{\boldsymbol{s}}_t^{p,k+1} - \hat{\boldsymbol{s}}_t^{p,k}) \oplus \boldsymbol{s}_t^i), \tag{9}$$

where m is the total number of resolutions and k indexes the resolution. $f_{\phi_0}(\cdot)$ and $f_{\phi_k}(\cdot)$ represent the base and k-th residual policies. All results of the policies are accumulated to obtain the final action $\boldsymbol{a}_t$.

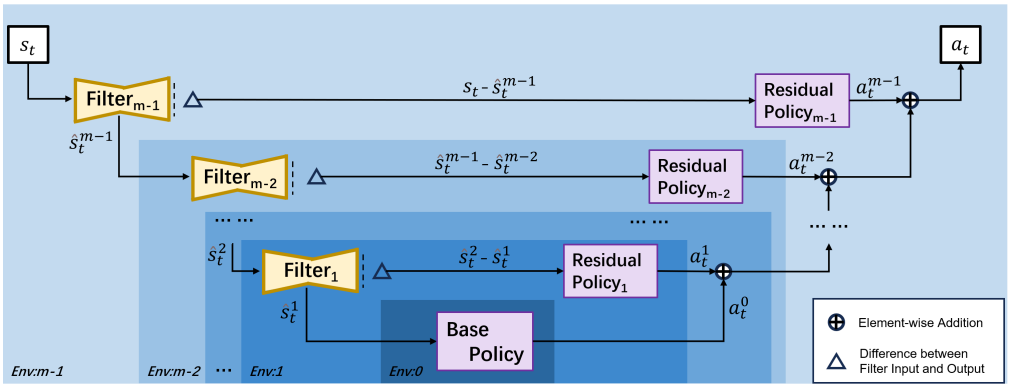

Fig. 2. **Structure of our multi-level policy network.** As mentioned in the main texts, $\boldsymbol{s}_t$ can be decomposed into $\boldsymbol{s}_t^p$ and $\boldsymbol{s}_t^i$, where only $\boldsymbol{s}_t^p$ needs to be processed. For simplification, $\boldsymbol{s}_t$ represents $\boldsymbol{s}_t^p$ only and $\boldsymbol{s}_t^i$ is fed to all policy blocks. Training begins from darker areas, representing simulations with lower resolutions, and proceeds to lighter ones, adding new modules progressively. Finally, results from all policies are accumulated as output action $\boldsymbol{a}_t$.

The proposed multi-level policy enables a progressive and faster reinforcement learning for control policy in simulation. However, with the augmentation of resolutions, a pertinent issue emerges: the addition of new filter modules can result in the accumulation and escalation of filter losses, potentially impeding the parameter updates of these new filters. To mitigate this challenge, it becomes imperative to constrain the loss function to ensure uniform scaling and facilitate the training of new filters. Consequently, we introduce a discount factor denoted as γ for ascending resolutions in the filter loss, serving to cap the upper bound of the loss and sustain consistent training conditions for the new filters:

$$\begin{aligned} J_F(\delta) &= \sum_{k=1}^{m-1} \gamma^{k-1} \|\hat{\boldsymbol{s}}_t^{p,k+1} - \hat{\boldsymbol{s}}_t^{p,k}\|_2^2, \\ &= \sum_{k=1}^{m-1} \gamma^{k-1} \|\hat{\boldsymbol{s}}_t^{p,k+1} - F(\hat{\boldsymbol{s}}_t^{p,k+1}|\delta_k)\|_2^2 \ , \end{aligned} \tag{10}$$

where $\gamma \in [0, 1]$ and in this paper we always used $\gamma = 0.5$.

D. Progressive training

Given our utilization of the SAC method [11] for training purposes in this paper, we explore additional strategies to enhance its efficiency. Specifically, experiences stored in the replay buffer at the same resolution can be reused across consecutive stages, thereby maintaining stability in the entropy coefficient, as these experiences are cleared upon resolution switches. Similarly, to mitigate interference from short-term unstable actions generated by the new filter, the critic component can remain fixed. Consequently, the loss function based on Eq. (2) undergoes modification as follows.

$$J_\pi(\phi, \delta) = \mathbb{E}_{\hat{\boldsymbol{s}}_t \sim \mathcal{D}} \left[D_{KL} \left(\pi_{\phi,\delta}^{new}(\cdot|\hat{\boldsymbol{s}}_t) \middle|\middle| \frac{\exp(Q_\theta^{old}(\hat{\boldsymbol{s}}_t, \cdot))}{Z_\theta^{old}(\hat{\boldsymbol{s}}_t)} \right) \right], \tag{11}$$

where the critic's Q-value network Q_θ^{old} inherits from the lower resolution one, and all input states should be first filtered into the estimated ones $\hat{\boldsymbol{s}}_t$. The approach above can limit the negative effects of new modules and keep critic modules adapted, leading to a more robust policy with as less parameters to be updated at each step as possible. Note that for standardization of module reuse, all undercomplete filter or policy modules share the same structure size, and the dimension size of states across simulations should remain the same, as only proprioceptive resolution-relevant states are always filtered. The whole policy network structure for multi-level training is shown in Fig. 2, and the general procedure of the algorithm is presented in Algorithm 1.

Algorithm 1 Our multi-level progressive RL algorithm

```
Initialize base policy φ_0 and critic ψ;
Train base policy φ_0 and critic ψ in initial resolution;
for m increasing higher resolutions do
    Fix all previous policies φ_i, filters δ_j and the critic ψ,
    where i ∈ [0, m − 1] and j ∈ [1, m − 1];
    Train filter δ_m with regularization objective;
    Fix all previous policies φ_i and filters δ_j, where i ∈
    [0, m − 1] and j ∈ [1, m];
    Train critic ψ and residual policy φ_m;
end for
```

V. Results and Validations

To present our results and validate our method, we applied our approach for policy learning in two fluid-solid coupled simulations, which are computationally costly. Comparisons are made to show that our method reduces training time apparently while retaining similar control policy.

A. Selection of tasks

We selected two typical fluid-solid coupled simulations to test our algorithm. In the first test, a 2D jet flow control problem was considered, which has long been recognized as a challenging task for its aerodynamic instability. By constructing a box-shaped aerial robot in 2D with controllable jet direction and velocity, vortices can be generated around the robot boundary, which is an intricate problem for motion control. We call this control task the Jet Flow Control (JFC) task. In the second test, we used FishGym [33], a GPU-based 3D simulator for training fish-like robots, which has higher

TABLE I

TRAINING PARAMETERS

r_{al}	1	w_a	-2e1	w_{ie}	-6e-2
w_{sm}	-1	w_{st}	1	w_e	-2e-3
dim. of $\mathbf{s}_t$	20	dim. of $\boldsymbol{a}_t$	2	batch size	256
ω_F	3e3	learning rate	1e-3	l_w	8

complexity since the fish body is formed by an articulated skeleton, and we call this control task as FishGym task. Evaluations will be performed on these two tasks in order to verify the robustness and effectiveness of our method.

B. Experimental setup

Since we employed the same setup for FishGym test as in [33], we will mainly introduce setup in JFC task instead.

1) Description of JFC task: In JFC task, the box-shaped robot aims to stabilize its desired orientation by controlling the jet flow. The reward function is defined below:

$$r = r_{al} + w_a r_a + w_{ie} r_{ie} + w_e r_e + w_{sm} r_{sm} + w_{st} r_{st}, \quad (12)$$

where w is the tunable weight for each term; r_{al} is a positive constant reward encouraging longer control; $r_a = -\|\alpha - \alpha_{goal}\|_2^2$ is the mean squared error between the desired and current attitudes; $r_{ie} = -\sum_t \gamma_{ie}^t \cdot r_a$ is the sum of integral error for angles in an episode with a discount factor γ; $r_e = -v_e^2$ is an energy term where v_e is the flow speed at the jet outlet; r_{st} is constant if the angle of the robot is stabilized in the vicinity of the target angle; and $r_{sm} = -\|\boldsymbol{a}_t - \boldsymbol{a}_{t-1}\|_2^2$ punishes sudden changes in actions to prevent potential servo failures in reality. All these parameters are listed in Table. I. The observations $\boldsymbol{o}_t$ in the JFC task contain the current angle α_t and angular velocity w_t of the robot itself. The action of the robot $\boldsymbol{a}_t$ contains its jet flow direction and speed.

2) Simulation parameters: The air density in the JFC task simulator is set as $1.225\ kg/m^3$, and viscosity is set to $1.81 \times 10^{-5} kg/(m \cdot s)$. The domain size in the simulator is set as $1m \times 2m$, and when it is simulated using the highest resolution, the domain contains 1000×2000 grid cells. The basic control step duration is $0.01s$, which would cost approximately $0.72s$ in computations. As the current solver places high demands on both CPU and GPU performance, we conducted all simulations using an AMD R5-3800X CPU and an NVidia RTX 3090 GPU.

C. Training

The policy network can be trained by employing SAC [10], [11] due to sampling efficiency and robust performance; however, it can be slow if it is trained using the original high-resolution simulation only. Alternatively, we can train the policy network in a coarse-to-fine manner: with initial random parameters, we can first train in a simulation with the coarsest resolution; after convergence, we use the trained results as initialization and proceed training on the higher resolution until we reach the final resolution. We name this training process as SAC with multi-resolution (MR), which can improve training efficiency, but may subject to unreliable convergence due to larger parameter space. Our module-based progressive training enables learning corresponding modules that match the respective simulations, reducing the parameter spaces while achieving more reliable convergence. Since training is significantly faster than simulation, we perform training on CPU instead.

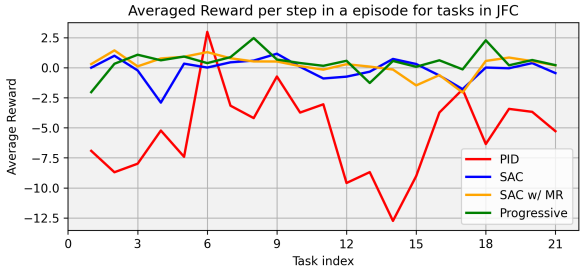

Fig. 3. **Averaged reward per step in tasks of JFC.** The figure above shows the averaged reward of steps over different tasks in JFC test, which is better if the value is higher. For most of the tasks, our progressive training method can achieve similar or even better results than other methods.

D. Evaluation

We assess the effectiveness of the proposed method by comparing the control performance in JFC task with that of a classic PID control and of the results using other training methods, such as SAC and SAC with MR. Additionally, we examine the training time and efficiency, both in the JFC and FishGym tasks. To confirm the effectiveness and necessity of the proposed method, we conducted ablation tests on selected modules within our framework.

1) Performance in JFC task: In this section, we compare the results with PID control and our policy network trained by SAC and SAC with MR, where PID parameters have been already tuned manually. The training process is evaluated by 21 distinct attitude tracking tasks. Performance comparisons were performed across various metrics, such as the averaged reward per episode, the number of stable duration per episode, the overall success ratio, and the total training time. Stable duration refers to the time interval during which the robot's angle remains close to the target within one degree. The criteria to judge the success of a task is to check whether the agent can keep stability for at least 10 control time units, which is $0.1s$ in simulation. As shown in Fig. 3, all learning-based methods exhibit much improved averaged reward and stability compared to PID control. Conversely, training incorporating a time window of observations as input exhibit better performance. Furthermore, by examining Table. II where the best results are highlighted in bold, we observe that our policy network with progressive training outperforms the other two training methods while requiring much less time.

2) Ablation study: We also conducted ablation experiments to validate the effectiveness of various modules incorporated in our RL framework, such as the filter and residual policy modules in the same resolution. As shown in Table. III, the model trained in a lower resolution, which receives coarser observations, exhibits inferior performance in the test

TABLE II

EVALUATIONS OF PERFORMANCE

Models	Avg. Reward	# of pre-exits	Stable duration	Success Ratio	Training time
PID	-5.356	6	52.095	47.72%	-
SAC [11]	-0.142	1	56.429	71.43%	212.8 hrs
SAC w/ MR	0.231	**0**	37.238	42.86%	158.3 hrs
Progressive	**0.458**	**0**	**111.905**	**76.19%**	**114.4 hrs**

TABLE III

ABLATION EVALUATIONS

Models	Filter	Residual Policy	Avg. Reward	Stable duration	Success Ratio
SAC [11]	w/o	w/o	-0.142	56.429	71.43%
Our method at step 1	w/o	w/o	-0.092	42.476	66.67%
Our method at step 2	w/	w/o	0.267	61.714	**76.19%**
Our method at step 3	w/	w/	**0.458**	**111.91**	**76.19%**

conditions. The difference in control capability also serves to confirm the gap between different simulation environments. By implementing a filter to bridge the gap and enhance the robustness of the controller as well as introducing the residual policy module to compensate for discrepancies in resolutions, our proposed method succeeds to improve the agent's control capability.

3) Training efficiency: We tested our training efficiency using both JFC and FishGym tasks, as shown in Figs. 4 and 5, respectively. In Fig. 4, both training convergence using SAC and SAC with MR are compared with our progressive approach, where different steps indicate training using different resolutions. It is clear that our progressive training can reach similar performance stably with much less time: our progressive method achieves time savings of 46.2% compared to SAC [11] and 27.3% compared to SAC with MR. Note that the number of steps at each stage is fixed to ensure consistency in comparison, and that SAC with MR lacks efficient re-utilization of past experiences, resulting in a gradual decline in performance after each resolution switch, leading to potential performance loss due to insensitivity to feedback among different resolutions. Similarly, Fig. 5 shows that our progressive training method exhibits a significantly faster convergence compared to SAC, resulting in approximately a 57.1% reduction in training time. This result affirms again the effectiveness of the progressive training method in a more complex 3D fluid-solid coupled simulation environment, indicating a potential for enhancing training efficiency in other complex simulations.

VI. CONCLUSION

In this paper, we present multi-level progressive reinforcement learning as the first trial to expedite the training process in a computationally expensive simulation, such as high-resolution fluid-solid coupled simulations. As previously mentioned, a novel multi-level framework involving filter and residual policy modules is proposed, which effectively address discrepancies observed across different resolutions.

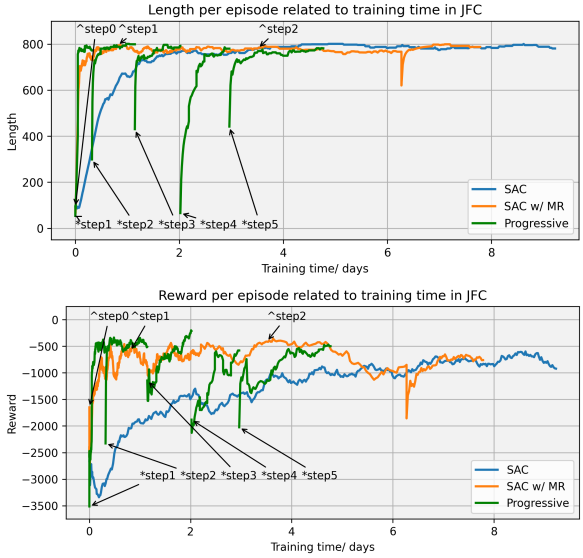

Fig. 4. **Training performance for JFC task.** All experiments were completed within a week, which is a reasonable time considering the fidelity of the simulation. The averaged reward curves indicate that our progress training has a more reliable convergence. Each time the SAC with MR switches resolution or our progressive method adds modules, there is an arrow pointing to the switching point with ^step and *step, respectively.

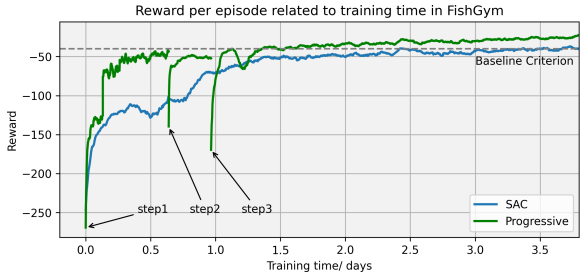

Fig. 5. **Training performance for FishGym task.** We used a gray dashed line to mark the criterion that the SAC reached in 4 days. There were two resolutions used in this experiment and the start of each step in our training stage is marked with an arrow in the figure. Overall, our progressive training method converges faster than the basic SAC method.

Our study verifies the practicality of employing progressively pre-trained filters and residual policy blocks to bridge the gap between different simulation resolutions, which achieves comparable, and in some cases superior, control performance compared to existing approaches, while substantially reducing training time. This reduction in training time holds significant importance in simulations that entail high computational costs.

ACKNOWLEDGEMENT

The authors would like to thank anonymous reviewers for helping improve exposition, as well as Wenbin Song, Zike Xu and Mengyun Liu from FLARE lab of ShanghaiTech University to help with numerical experiments. This work was supported by the National Science Foundation of China (No. 62072310) and ShanghaiTech University.

REFERENCES

[1] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.

[2] J. Allard, S. Cotin, F. Faure, P.-J. Bensoussan, F. Poyer, C. Duriez, H. Delingette, and L. Grisoni, "SOFA - an Open Source Framework for Medical Simulation," in *MMVR 15-Medicine Meets Virtual Reality*, vol. 125. IOP Press, 2007, pp. 13–18.

[3] H. Jasak, A. Jemcov, Z. Tukovic, *et al.*, "OpenFOAM: A C++ library for complex physics simulations," in *International workshop on coupled methods in numerical dynamics*, vol. 1000, 2007, pp. 1–20.

[4] S. Min, J. Won, S. Lee, J. Park, and J. Lee, "SoftCon: simulation and control of soft-bodied animals with biomimetic actuators," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–12, 2019.

[5] T. Zhang, R. Tian, H. Yang, C. Wang, J. Sun, S. Zhang, and G. Xie, "From simulation to reality: A learning framework for fish-like robots to perform control tasks," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3861–3878, 2022.

[6] W. Li, Y. Chen, M. Desbrun, C. Zheng, and X. Liu, "Fast and Scalable Turbulent Flow Simulation with Two-Way Coupling," *ACM Transactions on Graphics*, vol. 39, no. 4, pp. Art–No, 2020.

[7] Y. Chen, W. Li, R. Fan, and X. Liu, "GPU optimization for high-quality kinetic fluid simulation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 9, pp. 3235–3251, 2021.

[8] B. Ren, X. Ye, Z. Pan, and T. Zhang, "Versatile Control of Fluid-directed Solid Objects Using Multi-task Reinforcement Learning," *ACM Transactions on Graphics*, vol. 42, no. 2, pp. 1–14, 2022.

[9] M. Müller, "Hierarchical position based dynamics," 2008.

[10] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[11] 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.

[12] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv preprint arXiv:1606.01540*, 2016.

[13] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles," in *Field and Service Robotics: Results of the 11th International Conference*. Springer, 2018, pp. 621–635.

[14] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning," *arXiv preprint arXiv:2108.10470*, 2021.

[15] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "DroNet: Learning to Fly by Driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[16] T. Hu, K. Low, L. Shen, and X. Xu, "Effective phase tracking for bioinspired undulations of robotic fish models: A learning control approach," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 1, pp. 191–200, 2012.

[17] J. Lorenzetti, A. R. McClellan, C. Farhat, and M. Pavone, "UAV aircraft carrier landing using CFD-based model predictive control," in *AIAA Scitech 2020 Forum*, 2020, p. 1721.

[18] X. He, Z. Zhao, R. Ma, N. Wang, and L. Zhang, "Validation of hyperflow in subsonic and transonic flow," *Acta Aerodynamica Sinica*, vol. 34, no. 2, pp. 267–275, 2016.

[19] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 7559–7566.

[20] R. Julian, B. Swanson, G. S. Sukhatme, S. Levine, C. Finn, and K. Hausman, "Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning," *arXiv preprint arXiv:2004.10190*, 2020.

[21] V. Campos, P. Sprechmann, S. Hansen, A. Barreto, S. Kapturowski, A. Vitvitskyi, A. P. Badia, and C. Blundell, "Beyond fine-tuning: Transferring behavior in reinforcement learning," *arXiv preprint arXiv:2102.13515*, 2021.

[22] 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.

[23] A. Gruslys, W. Dabney, M. G. Azar, B. Piot, M. Bellemare, and R. Munos, "The reactor: A fast and sample-efficient actor-critic agent for reinforcement learning," *arXiv preprint arXiv:1704.04651*, 2017.

[24] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, *et al.*, "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures," in *International conference on machine learning*. PMLR, 2018, pp. 1407–1416.

[25] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver, "DISTRIBUTED PRIORITIZED EXPERIENCE REPLAY," *arXiv preprint arXiv:1803.00933*, 2018.

[26] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive Neural Networks," *arXiv preprint arXiv:1606.04671*, 2016.

[27] J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox, "GPU-accelerated robotic simulation for distributed reinforcement learning," in *Conference on Robot Learning*. PMLR, 2018, pp. 270–282.

[28] J. Rabault and A. Kuhnle, "Accelerating deep reinforcement learning strategies of flow control through a multi-environment approach," *Physics of Fluids*, vol. 31, no. 9, 2019.

[29] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[30] B. D. Ziebart, *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

[31] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[33] W. Liu, K. Bai, X. He, S. Song, C. Zheng, and X. Liu, "FishGym: A High-Performance Physics-based Simulation Framework for Underwater Robot Learning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6268–6275.