



Fig. 4. Baselines of **ASAP**. (a) Model-free RL training. (b) System ID from real to sim using real-world data. (c) Learning delta dynamics model using real-world data. (d) Our proposed method, learning delta action model using real-world data.

physics. We leverage an RL-based delta/residual action model to compensate for the sim-to-real physics gap.

As illustrated in Figure 2 (b), the delta action model is defined as $\Delta a_t = \pi_\theta^\Delta(s_t, a_t)$, where the policy π_θ^Δ learns to output corrective actions based on the current state s_t and the action a_t . These corrective actions (Δa_t) are added to the real-world recorded actions (a_t^r) to account for discrepancies between simulation and real-world dynamics.

The RL environment incorporates this delta action model by modifying the simulator dynamics as follows: $s_{t+1} = f^{\text{sim}}(s_t, a_t^r + \Delta a_t)$ where f^{sim} represents the simulator’s dynamics, a_t^r is the reference action recorded from real-world rollouts, and Δa_t introduces corrections learned by the delta action model.

TABLE II
REWARD TERMS FOR DELTA ACTION LEARNING

Term	Weight	Term	Weight
Penalty			
DoF position limits	-10.0	DoF velocity limits	-5.0
Torque limits	-0.1	Termination	-200.0
Regularization			
Action rate	-0.01	Action norm	-0.2
Task Reward			
Body position	1.0	VR 3-point	1.0
Body position (feet)	1.0	Body rotation	0.5
Body angular velocity	0.5	Body velocity	0.5
DoF position	0.5	DoF velocity	0.5

During each RL step:

- 1) The robot is initialized at the real-world state s_t^r .
- 2) A reward signal is computed to minimize the discrepancy between the simulated state s_{t+1} and the recorded real-world state s_{t+1}^r , with an additional action magnitude regularization term $\exp(-\|a_t\|) - 1$, as specified in Table II. The workflow is illustrated in Figure 2 (b).
- 3) PPO is used to train the delta action policy π_θ^Δ , learning corrected Δa_t to match simulation and the real world.

By learning the delta action model, the simulator can accurately reproduce real-world failures. For example, consider a scenario where the simulated robot can jump because its motor strength is overestimated, but the real-world robot cannot jump due to weaker motors. The delta action model π_θ^Δ will learn to reduce the intensity of lower-body actions, simulating the motor limitations of the real-world robot. This allows the simulator to replicate the real-world dynamics and enables the policy to be fine-tuned to handle these limitations effectively.

C. Fine-tuning Motion Tracking Policy under New Dynamics

With the learned delta action model $\pi^\Delta(s_t, a_t)$, we can reconstruct the simulation environment with

$$s_{t+1} = f^{\text{ASAP}}(s_t, a_t) = f^{\text{sim}}(s_t, a_t + \pi^\Delta(s_t, a_t)),$$

As shown in Figure 2 (c), we keep the π^Δ model parameters frozen, and fine-tune the pretrained policy with the same reward summarized in Table I.

D. Policy Deployment

Finally, we deploy the fine-tuned policy without delta action model in the real world as shown in Figure 2 (d). The fine-tuned policy shows enhanced real-world motion tracking performance compared to the pre-trained policy. Quantitative improvements will be discussed in Section IV.

IV. PERFORMANCE EVALUATION OF **ASAP**

In this section, we present extensive experimental results on three policy transfers: IsaacGym [58] to IsaacSim [63], IsaacGym to Genesis [6], and IsaacGym to real-world Unitree G1 humanoid robot. Our experiments aim to address the following key questions:

- **Q1:** Can **ASAP** outperform other baseline methods to compensate for the dynamics mismatch?
- **Q2:** Can **ASAP** finetune policy to outperform SysID and Delta Dynamics methods?
- **Q3:** Does **ASAP** work for sim-to-real transfer?

Experiments Setup. To address these questions, we evaluate **ASAP** on motion tracking tasks in both simulation (Section IV-A and Section IV-B) and real-world settings (Section IV-C).

In the simulation, we use the retargeted motion dataset from the videos we shoot, denoted as $\mathcal{D}_{\text{Robot}}^{\text{Cleaned}}$, which contains diverse human motion sequences. We select 43 motions categorized into three difficulty levels: easy, medium, and hard (as partially visualized in Figure 6), based on motion complexity and the required agility. **ASAP** is evaluated through simulation-to-simulation transfer by training policies in IsaacGym and using two other simulators, IsaacSim and Genesis, as a proxy of “real-world” environments. This setup allows for a systematic evaluation of **ASAP**’s generalization and transferability. The success of the transfer is assessed by metrics described in subsequent sections.

For real-world evaluation, we deploy **ASAP** on Unitree G1 robot with fixed wrists to track motion sequences that has obvious sim-to-real gap. These sequences are chosen to