SWIM: COMPACT ENVIRONMENT REPRESENTATION FOR REINFORCEMENT LEARNING HUMAN SWIMMING

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep reinforcement learning (RL) has driven rapid progress in physically-based motion generation, yet synthesizing robust motion policies in dense fluid environments (*e.g.* swimming) remains unsolved. Unlike land motion, where environmental impact is sparse and can be coarsely modeled (*e.g.* gravity, normal reaction), swimming requires continuous, full-body coordination under pressure and flow forces across the entire body surface; fully-coupled rigid–fluid simulation is far too slow for the millions of interactions RL requires. We propose SWIM, an RL framework for physically-based human swimming learned from a single reference motion. Its core is a *compact, low-dimensional environment representation:* a dual-branch graph-convolutional VQ-VAE that tokenizes per-link body–water forces and torques, informative enough for control yet robust to the rapidly changing force exchanges that would otherwise destabilize training. We pair this with a GPU-based Lagrangian solver with rigid–fluid coupling that yields 10–15× faster training. Across hundreds of training runs and several hundred zero-shot conditions, SWIM generalizes to unseen goals and trajectories, fluids with varying density, external flows and perturbations, altered body morphologies, and all four competitive swimming styles. Against imitation-learning baselines (MimicKit-DeepMimic, MimicKit-AMP, ADD) and alternative force models (a PINN world model, MuJoCo's inertial fluid model, and an underwater-robotics simulator), SWIM achieves better stability, goal satisfaction, and physical realism: *a 63.9% success rate on held-out generalization versus 36.8% for the strongest baseline*.

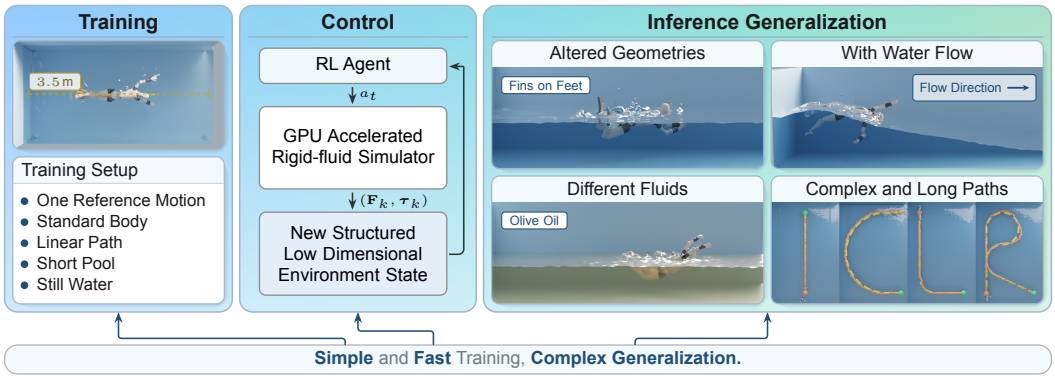

Figure 1: Overview of SWIM. Trained on a single simple goal-reaching task in still water (left), SWIM learns a policy (middle) that can generalize to significantly different tasks, fluid environments, perturbations, body geometries, across multiple swimming styles (right).

1 INTRODUCTION

Synthesizing physically-based human motion is a key challenge spanning machine learning, control theory, robotics, and now embodied AI. The primary goal is to learn robust and generalizable control policies that adhere to physical laws, while reacting naturally to environmental variations. Recently,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

reinforcement learning (RL) has made impressive progress in increasingly complex environments. Existing methods can now control virtual humans/agents to perform dynamic skills, traverse uneven terrains, and interact with objects (Zhang et al., 2023; Tessler et al., 2024; Wang et al., 2025; 2026). Much of this success comes from advances in fast simulation, state representations, and optimized training schemes. However, these successes are predominantly confined to land scenarios where distributed environmental impact can be ignored or coarsely modeled (e.g., gravity).

We push the complexity of environments by investigating physically-based control for swimming. Swimming, by its nature, introduces a challenging paradigm of dense character-environment interaction. The (control) agent must dynamically apply forces to the environment to generate reactive propulsion and successfully swim (forward). However, unlike simple ground contact (*i.e*. contact force) in land motion, the aquatic medium continuously applies pressure and fluid forces across the entire body surface. These environmental forces are dynamic, frequently shifting in magnitude and direction even in still water conditioned on the style/type of swimming, let alone in dynamic flows and waves. Mastering this full-body coordination requires years of training for humans (*e.g*. to stay afloat, swim, and/or move towards a target). Learning such a control policy requires a notable leap in handling task complexity. Previously, optimization was employed by a few for swimming control (Yang et al., 2004; Si et al., 2014) but merely to produce prescribed motions (*e.g*. motion capture data). Learning control generalizable to unseen tasks/environments remains an open challenge.

The primary challenge in extending RL to swimming requires efficiently and effectively modeling dynamic environmental forces. While standard fully-coupled rigid-fluid simulations provide accurate body-water forces necessary for control learning, they are prohibitively slow for RL that requires millions of environment interactions during training (Gissler et al., 2019; Takahashi & Batty, 2022; 2020; Fang et al., 2020). On the other hand, while fast surrogate models exist (Vaxenburg et al., 2025; Sanchez-Gonzalez et al., 2020; Song et al., 2025a; Dou et al., 2025), they often fail to capture complex, full-body body-fluid interactions necessary for continuous swimming motion. An immediately following challenge is the high-dimensionality of the environment, *i.e*. water, applying rapidly and drastically changing forces continuously on the body surface. Since the water-body environment must be discretized and captured in RL states for learning, suitable representations are needed to balance physical fidelity (*e.g*. water-body force accuracy) and learning efficiency. Finally, similar to land motions, learning human-like motions requires data guidance as RL can 'cheat' for rewards by learning unnatural motions. However, unlike land motions, high-quality full-body motion capture data for swimming remains scarce, making data-hungry imitation methods infeasible.

To address the above challenges, we propose SWIM (Compact Environment Representation for RL Human Swimming). Our core contribution is a novel approach to model fluid-character dynamics that strikes a balance among physical fidelity, computational efficiency, and generalization. First, we introduce a low-dimensional state representation that effectively distills complex character-fluid interactions. This representation is sufficiently informative regarding local contact forces and the near-body fluid environment; at the same time, it is robust to the quickly changing water-body force exchanges. Second, to make training tractable, we redesign the learning loop with an optimized, GPU-based Lagrangian fluid simulator integrated with an articulated-body solver. This achieves upwards of $15\times$ speedup in training, tackling the simulation bottleneck. Together, these force-modeling advancements enable SWIM to learn robust control policies from a single reference motion (containing only local joint kinematics) and generalize them broadly. See Figure 1.

We evaluate SWIM across diverse tasks, environments, and swimming styles using more than 250 RL training runs and 80k zero-shot generalization settings spanning over diverse environmental dimensions (*e.g*. fluid density/viscosity, dynamic water speed/direction, external perturbations, and morphological changes). Compared to baselines adapting human imitation learning and surrogate models, SWIM demonstrates superior stability, robustness, and physical realism. Our key contributions are: (i) To the best of our knowledge, SWIM is the first method to learn stable, controllable, and natural full-body swimming policies with deep RL, from only a *single* reference motion. (ii) We introduce a structured state that tokenizes per-link hydrodynamic forces and torques via a dual-branch graph-convolutional VQ-VAE, informative enough for control yet robust to the rapidly changing force exchanges that would otherwise destabilize training. (iii) Across hundreds of training runs and several hundred zero-shot conditions, SWIM generalizes to unseen goals, trajectories, fluids, flows, perturbations, morphologies, and all four competitive strokes, achieving a 63.9% success rate on held-out generalization tasks versus 36.8% for the strongest baseline. In addition, we also contribute a non-trivial GPU implementation of a fast and stable rigid-fluid simulator.

2 Related Work

The overall positioning of our work is shown in Table 3. RL-based methods have demonstrated strong performance in locomotion and agile behaviors (Peng et al., 2018; Liu & Hodgins, 2017; Yang et al., 2022), and have been extended to more complex tasks such as sports (Zhang et al., 2023; Wang et al., 2025), fine manipulation (Yang et al., 2022; Xu et al., 2026), and large-scale motion modeling (Tessler et al., 2024; Yao et al., 2022; 2023). However, they typically assume simple environments with sparse interactions (*e.g.* motion under gravity). Swimming represents a significantly more challenging setting that involves continuous full-body interaction with a dynamic fluid environment. Early methods rely on hand-designed controllers and optimization (Tan et al., 2011; Si et al., 2014; Lentine et al., 2011; Kwatra et al., 2010; Yang et al., 2004), which require extensive tuning and lack generalizability. Recent work has started to investigate body–water interactions (Dou et al., 2025) but only on adapting motions in simple body-water interaction. Our research is the first RL-based method for physically-based character swimming motions, to the best of our knowledge.

Table 1: Comparison with literature.

Representative work	Humanoid	RL-based	Swimming
Yang et al. (2004)	✓	✗	✓
Kwatra et al. (2010)	✓	✗	✓
Tan et al. (2011)	✗	✗	✓
Si et al. (2014)	✓	✗	✓
Peng et al. (2018)	✓	✓	✗
Vaxenburg et al. (2025)	✗	✓	✗
Chu et al. (2025)	✗	✓	✓
Song et al. (2025b)	✗	✓	✓
SWIM(Ours)	✓	✓	✓

In fluid simulation, Eulerian methods provide high accuracy but are often prohibitively slow for RL learning (Takahashi & Batty, 2020; 2022; Wang et al., 2024). In contrast, Lagrangian approaches are fast but comparatively less accurate in computing force exchanges between rigid bodies and fluid (Koschier et al., 2019; Wang et al., 2024). Recently, neural network approaches (Jain et al., 2024; Raissi et al., 2019) handle rigid-fluid coupling but their accuracy and generalization remain challenging (Thuerey et al., 2021; McGreivy & Hakim, 2024). In our research, Lagrangian methods provide the flexibility in balancing accuracy and speed for learning. We adopt and provide a non-trivial GPU implementation of DFSPH (Bender & Koschier, 2015) with stable rigid–fluid coupling (Akinci et al., 2012).

Control in fluid environments has been studied in underwater robots (Du et al., 2021; Chen et al., 2022), marine vehicles (Chu et al., 2025; Song et al., 2025a), and biological locomotion (Satoi et al., 2016), as well as differentiable fluid–rigid systems (Li et al., 2023). However, these systems typically involve low degree-of-freedom bodies or simplified interactions. In contrast, we address full-body control under complex and dynamic fluid conditions with high degree-of-freedom bodies.

3 Methodology

3.1 Problem Formulation

We formulate swimming control as a Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \pi)$. Given an initial state $s_0 \in \mathcal{S}$, which describes the character, the environment, and the task, the character chooses an action $a_t \in \mathcal{A}$ at time t based on a policy $\pi(a_t \mid s_t)$. During the process, the state is updated by a transition function $\mathcal{T}(s_{t+1} \mid s_t, a_t)$. At time t, we can evaluate a reward r_t via $\mathcal{R}(a_t, s_t, s_{t+1})$. The objective is to learn the policy π that maximizes $J(\pi) = \mathbb{E}_\pi[\sum_t \gamma^t r_t]$. where $\gamma \in (0, 1]$ is the discount factor that determines the relative importance of immediate versus future rewards. Next, we detail $\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \pi$ and learning algorithm. More details are in the Appendix C.

3.2 State Space for Body-Water Interaction

The state $s_t = [s_t^{char}, s_t^{goal}, s_t^{env}]$ consists of body kinematics s_t^{char}, the goal s_t^{goal}, and the surrounding environment s_t^{env}. Our character is an articulated humanoid with 36 controllable degrees of freedom (DoFs) and a 6-DoF root (Figure 3 left). Similar to prior research (Zhang et al., 2025), $s_t^{\text{char}} \in \mathbb{R}^{174}$ includes joint position, angle, velocity, angular velocity. Also, we include two variables to parameterize stroke cycles: a phase and a frequency (as explained in Section 3.4), to disambiguate recurring poses in different cycles and motion frequency. More details are in the Appendix C.1.1.

We propose a multi-horizon goal representation $s_t^{\text{goal}} \in \mathbb{R}^{12}$, with an example shown in Figure 2:

$$s_t^{\text{goal}} = \left[g_1(t)^\top, g_2(t)^\top, g_3(t)^\top, g_4(t)^\top\right]^\top \in \mathbb{R}^{12}, \tag{1}$$

where $g_i(t) = [x_i^{\text{loc}}, y_i^{\text{loc}}, \psi_i^{\text{loc}}]^\top$ represents the ith waypoint parameterized by its location $[x_i^{\text{loc}}, y_i^{\text{loc}}]$ in the x-y plane (z axis being the up direction) and local heading deviation ψ_i^{loc} toward the next waypoints at $\delta t_i \in \{0.1,\ 0.2,\ 0.8,\ 1.6\}$ s. More details are in the Figure 12 and Appendix A.7. The multi-horizon goal design is for balancing motion quality and controllability. Only aiming for the final destination provides limited guidance in the middle, leading to poor controllability, while overly dense control (*e.g.* control in every frame) is restrictive for the character to adapt to water dynamics. This is because swimming maneuvers such as steering can only be effectively executed in certain phases of a stroke cycle, *e.g.* steering in butterfly requires asymmetrical arm pulling which can only be effective when the upper body is in water. Forcing these maneuvers by dense control at best generates unnatural motions and at worst fails to swim at all.

3.3 Environment State

The biggest challenge of SWIM is the environment state representation which needs to balance between learning efficiency and stability, finding a sweet spot of how accurate the water-body force needs to be. Highly accurate forces require high-resolution solvers (*e.g.* Finite Element methods with ultra-fine meshes) that can accurately compute water-body coupling, which is prohibitively slow for RL. Conversely, surrogate models such as neural solvers are sufficiently faster but often lead to inaccurate force estimation, which affects learning, as shown later. Furthermore, regardless of the solver, body-water forces change rapidly in swimming, leading to unstable training. Therefore, we propose a representation that is sufficiently accurate, fast to compute, and changes reasonably smoothly.

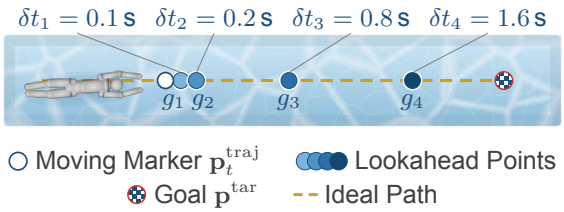

Figure 2: The goal state s_t^{goal} contains four lookahead waypoints $g_i(t)$ at time offsets δt_i.

Observing that in swimming the body-water force is only qualitatively sensed by our body, not exactly computed in our brain, we abstract the force in space and time in a compact and structured. We first divide the body into 13 links (Figure 3 left) and the environment state $s_t^{\text{env}} = [s_t^{\text{env,total}}, s_t^{\text{env,token}}]$ describes the whole-body wrench $s_t^{\text{env,total}}$ and individual body-water forces $s_t^{\text{env,token}}$. The whole-body wrench is the net force and net torque on the whole body:

$$s_t^{\text{env,total}} = \left[\mathbf{F}_{\text{total}}^\top,\ \boldsymbol{\tau}_{\text{total}}^\top\right]^\top, \quad \mathbf{F}_{\text{total}} = \sum_{k=1}^{13} \mathbf{F}_k, \quad \boldsymbol{\tau}_{\text{total}} = \sum_{k=1}^{13} \Big(\boldsymbol{\tau}_k + (\mathbf{c}_k - \mathbf{c}_{\text{root}}) \times \mathbf{F}_k\Big), \tag{2}$$

where $\mathbf{c}_k, \mathbf{c}_{\text{root}} \in \mathbb{R}^3$ denote the center of mass of body k and the root, respectively, with $\mathbf{F}_k \in \mathbb{R}^3$ and $\boldsymbol{\tau}_k \in \mathbb{R}^3$ being the force and torque acting on body k (Section 3.5). Next, the body-water forces are also represented for each body link as a compact representation of fluid interaction:

$$s_t^{\text{env,token}} = \left[\mathbf{z}_{1,t}^{F\top},\ \mathbf{z}_{1,t}^{T\top},\ \ldots,\ \mathbf{z}_{13,t}^{F\top},\ \mathbf{z}_{13,t}^{T\top}\right]^\top \in \mathbb{R}^{78}, \tag{3}$$

where $\mathbf{z}_{k,t}^F, \mathbf{z}_{k,t}^T \in \mathbb{R}^{d_z}$ are features representing the force and torque applied to body k by water. Note that $\mathbf{z}_{k,t}^F, \mathbf{z}_{k,t}^T$ are not raw forces but qualitative representations of body-water forces. For the considerations explained above, these representations need to mimic how humans feel (*i.e.* qualitative sensing) water dynamics around different body parts when swimming. In addition, they need to faithfully represent the raw forces but they themselves should not change too abruptly in time. To this end, in practice, we propose to tokenize individual body-water forces, via a dual-branch graph convolutional vector quantizer based on VQ-VAE (Van Den Oord et al., 2017), shown in Figure 3. Our limb-level tokenizer is built around two considerations. First, force and torque cause two types of motions in different spaces: linear and angular motions. We therefore employ two sub-networks with independent codebooks ($\mathcal{C}_F$ and $\mathcal{C}_T$, each containing $K = 512$ vectors of dimension $d_z = 3$), to capture quantized force and torque separately in a compact and stable form.

Second, both force and torque propagate on skeleton following a tree structure which is a directed graph from parent joints to child joints. This is essential as the dynamics of the parent link largely

determines that of the child link. As shown in (Appendix B.3), this one-way propagation is crucial in learning and zero-shot transfer. Specifically, the encoder (Figure 3 middle) maps raw forces and torques of individual bodies to latents, which are then quantized using their respective codebooks:

$$\mathbf{z}_k^F = \mathrm{VQ}_{\mathcal{C}_F}(\hat{\mathbf{z}}_k^F), \quad \mathbf{z}_k^T = \mathrm{VQ}_{\mathcal{C}_T}(\hat{\mathbf{z}}_k^T). \tag{4}$$

The words $\mathbf{z}_k^F$ and $\mathbf{z}_k^T$ are latent representations of water-body forces. We pre-train the encoder using self-supervised wrench reconstruction and freeze it during RL learning, so the control is learned in the latent space, while preserving the key physical information of the original wrench.

3.4 Policy and Action Space

The policy π is a neural network that maps a state s_t to an action a_t: $a_t = [\omega_t, \Delta\mathbf{q}_t] \in \mathbb{R}^{1+36}$, where ω_t is stroke frequency and $\Delta\mathbf{q}_t \in \mathbb{R}^{36}$ is a correction (Bergamin et al., 2019; Yuan & Kitani, 2020), both of which are used to compute the target of the control:

$$\begin{aligned} \phi_t &= \big(\phi_{t-1} + \omega_t \Delta t\big) \in [0, 1), \\ \mathbf{q}_t^{\text{target}} &= \mathbf{q}^{\text{ref}}(\phi_t) + \Delta\mathbf{q}_t, \end{aligned} \tag{5}$$

where Δt is the time step, $\mathbf{q}_t^{\text{target}}$ is the target pose for proportional-derivative (PD) control (Peng, 2025), ϕ_t is the phase, and $\mathbf{q}^{\text{ref}}(\phi_t)$ the pose sampled at the ϕ_t from a reference motion Section 4.1. To prevent abrupt changes, both action outputs are smoothed using first-order exponential moving average filters, with the smoothing parameter based on research on the frequency of muscle control (Crenna et al., 2021). The filtering is constrained within $\omega_t \in [0.5, 1.5]$. More details in the Appendix C.2.

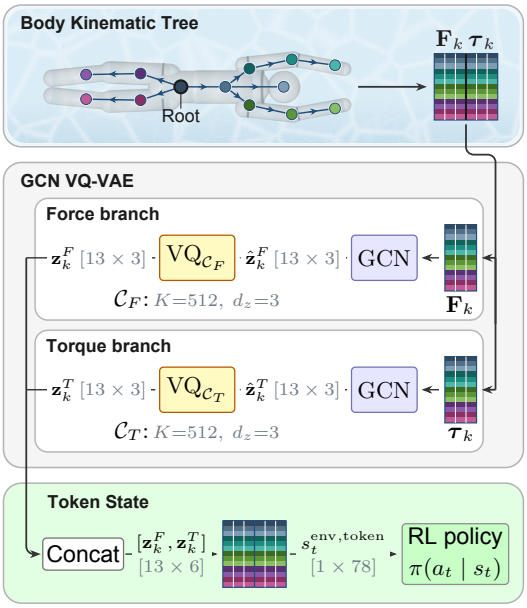

Figure 3: **Dual Directed-Branch Graph Convolutional Network VQ-VAE tokenizer.** Hydrodynamic forces $\mathbf{F}_k$ and torques $\boldsymbol{\tau}_k$ propagate from parent nodes to child nodes along the skeleton. Separate codebooks quantize $\mathcal{C}_F$ and $\mathcal{C}_T$, providing a final $s_t^{\text{env,token}} = [\mathbf{z}_k^F, \mathbf{z}_k^T]$ for RL learning.

3.5 Body-fluid Simulation as Transition Function

The transition function $\mathcal{T}(s_{t+1} \mid s_t, a_t)$ is realized as a physical simulator. We choose a Lagrangian method. Despite their general inferiority in accuracy to Eulerian methods, they strike a good balance between accuracy and speed. However, empirically we found many existing implementations are still slow. Therefore, we re-implemented the CPU-based SPlisHSPlasH simulator (Bender et al., 2017) with DFSPH (Bender & Koschier, 2015) and rigid-fluid coupling (Akinci et al., 2012) fully on GPU and gained 15× acceleration on a single NVIDIA GeForce RTX 5090 GPU. More details are in the Appendix C.3.

3.6 Swimming-Informed Reward Design

Our reward function $r_t = \mathcal{R}(a_t, s_t, s_{t+1})$ drives the character towards the goal by a task reward r_t^{task}, maintains body stability by a body stability reward r_t^{stab}, and preserves motion naturalness by a motion smoothness reward r_t^{accel}. Together, these terms provide complementary training signals. Similar to previous work (Mysore et al., 2021), we use a multiplicative-additive form:

$$r_t = r_t^{\text{task}} \cdot r_t^{\text{stab}} + w_{\text{accel}} \cdot r_t^{\text{accel}}. \tag{6}$$

We decompose the task into three complementary objectives. Given a target where we assume a straight path, or an arbitrary control trajectory, we consider the progress accomplished along the path, the lateral alignment of the body with the path, and the heading direction, as defined:

$$r_t^{\text{task}} = r_t^{\text{forward}} \cdot r_t^{\text{deviation}} \cdot r_t^{\text{heading}} = \exp\left(-\alpha_p d_t^{\text{forward}}\right) \exp\left(-\alpha_a d_t^{\text{deviation}}\right) \exp\left(-\alpha_h e_t^{\text{yaw}}\right)$$

The d_t^{forward}, $d_t^{\text{deviation}}$, and e_t^{yaw} denote the forward distance error, lateral deviation distance error, and heading-yaw error, respectively, as shown in Figure 4, detailed in Appendix C.4.

Swimming styles such as `freestyle` require large body roll (Psycharakis & Sanders, 2010), which during learning can easily cause unnatural motions, even flipping, of the body. To mitigate this, we design a stability reward: $r_t^{\text{stab}} = \exp\left(-\alpha_r \cdot d_t^{\text{roll}}\right)$. Where d_t^{roll} measures the error between the current body roll angle and the ideal roll angle ((Psycharakis & Sanders, 2010)). Following (Peng et al., 2018), we penalize the mean absolute joint acceleration to encourage temporally smooth motion and suppress jitter. Detailed formulations are provided in Appendix C.4.

Training design for single-instance imitation. We design only one goal-reaching task for training explained later, to avoid long training time due to the required physical simulation. Also, we do not have reward terms requiring the generated motions to strictly follow the reference motion. This is important as we only use one reference motion for learning, which does not provide sufficient samples to describe the feasible swimming motion space for the policy. Therefore, we only restrict the updates in Equation (5) and encourage similar dynamic profiles in r_t^{accel}.

Training. We employ Proximal Policy Optimization (PPO) (Schulman et al., 2017) for learning with several swimming-specific adaptions. More details are provided in Appendix C.5.

4 Experimental Results

Due to space limit of the main paper, we provide the overall evaluation settings and the main results here, and leave the full details in Appendix A. We will release the code and data upon acceptance.

4.1 Experimental Setup

Since SWIM is the first RL-based method for human swimming and has no existing benchmarking protocols to follow, we first define tasks, metrics, and training and generalization settings.

Task. We define two tasks: *goal-reaching* and *trajectory-following*. Goal-reaching is, given a starting configuration and a goal position, to test if the character can reach the goal at the specified time. Trajectory-following is to test if the character can follow a pre-defined trajectory.

Metrics. We use eight metrics into two groups for task completion and motion quality, respectively. The first group includes success rate (**SR**), survival rate (**SRV**), progress (**Prog**), positional error (**PS**), and lateral deviation (**LD**): **SR** measures if the task is successful in given time; **SRV** measures if character survives testing (*e.g.* not excessive rolling beyond 90° or not falling out of the pool); **Prog** measures the overall progress accomplished; **PS** measures the deviation of the character from a desired trajectory; **LD** measures lateral deviation from the desired trajectory. The desired trajectory for both **PS** and **LD** is the straight line from the initial root position to the goal in goal-reaching, and the pre-defined trajectory in trajectory-following. More details are provided in Appendix A.1. The second group includes **Roll**, **Vel** and **Accel**: **Roll** measures if body rolling exceeds the specified tolerance range over time. **Vel** and **Accel** measure the joint velocity difference between the generated and reference motion, and the joint acceleration of the generated motion itself, respectively. The definitions and justifications of the metrics are given in Appendix A.5. All comparison and ablation results are averaged over three runs under the same conditions for completeness and consistency.

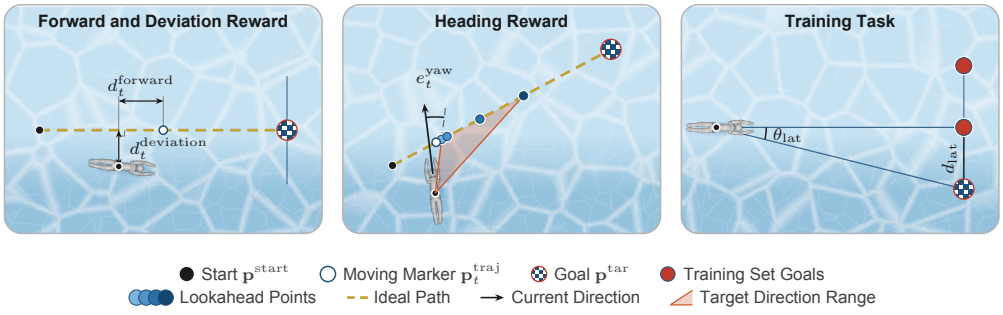

Figure 4: Left: forward and deviation reward. Middle: head reward. Right: training task. Red dots are goals. Blue dots are intermediate goals. Black dots are the initial position. The straight line between the initial position and a goal is the ideal trajectory.

Table 2: Results averaged across T×E conditions.

Task	Stroke	SRV↑ (%)	SR↑ (%)	PS↓ (m)	LD↓ (m)	Prog→ 1	Roll↓ (°)	Vel↓ (rad/s)	Accel↓ (rad/s^2)
Goal reaching	Front crawl	100.0	58.8	$0.223_{\pm 0.152}$	$0.102_{\pm 0.113}$	$0.856_{\pm 0.088}$	$1.02_{\pm 0.88}$	$1.04_{\pm 0.09}$	$54.92_{\pm 3.27}$
	Backstroke	99.4	32.5	$0.387_{\pm 0.331}$	$0.198_{\pm 0.201}$	$0.805_{\pm 0.180}$	$2.00_{\pm 2.65}$	$1.19_{\pm 0.04}$	$53.49_{\pm 2.23}$
	Breaststroke	100.0	62.5	$0.211_{\pm 0.221}$	$0.098_{\pm 0.166}$	$0.878_{\pm 0.123}$	$0.82_{\pm 0.72}$	$1.65_{\pm 0.08}$	$95.87_{\pm 10.04}$
	Butterfly	100.0	63.1	$0.195_{\pm 0.152}$	$0.089_{\pm 0.117}$	$0.877_{\pm 0.095}$	$0.54_{\pm 0.46}$	$1.58_{\pm 0.10}$	$76.82_{\pm 6.00}$
Trajectory following	Front crawl	99.4	49.4	$0.235_{\pm 0.139}$	$0.122_{\pm 0.119}$	$0.868_{\pm 0.080}$	$1.09_{\pm 1.85}$	$1.04_{\pm 0.08}$	$54.40_{\pm 3.21}$
	Backstroke	99.7	27.5	$0.321_{\pm 0.198}$	$0.191_{\pm 0.143}$	$0.863_{\pm 0.145}$	$2.25_{\pm 3.23}$	$1.18_{\pm 0.04}$	$53.08_{\pm 2.47}$
	Breaststroke	99.1	54.7	$0.210_{\pm 0.128}$	$0.109_{\pm 0.106}$	$0.911_{\pm 0.100}$	$0.97_{\pm 2.00}$	$1.65_{\pm 0.08}$	$96.67_{\pm 9.83}$
	Butterfly	100.0	59.1	$0.215_{\pm 0.154}$	$0.112_{\pm 0.127}$	$0.890_{\pm 0.083}$	$0.64_{\pm 0.72}$	$1.57_{\pm 0.09}$	$76.65_{\pm 5.51}$

Data. The only available full-body swimming motion dataset captured from a real human is the *Swimming Techniques* capture of the Porto Biomechanics Laboratory (LABIOMEP & Qualisys, 2025), a small dataset containing one instance for each style: `front-crawl`, `backstroke`, `breaststroke`, and `butterfly`, lasting only 6.44 s with only local joint-angle motion after preprocessing work. The body has 12 joints and the motion is sampled at 100Hz. During simulation, we divide the body surfaces into patches for water-body force estimation. Details are in Appendix A.

Training & Generalization. For each swimming style, we train one policy using a simple goal-reaching task with only 3 goals spanning $\theta_{lat} = 14°$ in a small pool (3.5m × 1.8m × 0.6m) in still water, shown in Figure 4 Right. For generalization tests, we use combinations of unseen environments including large pools (5.0m × 7.5m × 0.6 m and 8.0m × 8.5m × 0.6 m), goals, control trajectories, water flows, perturbations, body geometries and morphologies, and different fluids across a broad range of conditions. Note this is a strong generalization test as the training only involves short-distance swimming without control trajectories and in still water. Details are in Appendix A.

4.2 Strong Generalization to Unseen Conditions

We first report generalization results on exhaustive and challenging combinations of conditions in tasks, environments, perturbation, and body geometry and morphologies:

- (**T**) 10 sampled unseen goals up to $\theta_{lat} = 30°$ (2× wider than training), for goal-reaching; 20 unseen control trajectories containing short quick turns and slow long turns up to 30° represented by polylines and sinusoidal curves, for trajectory-following. A total of 30 tasks.
- (**E**) Water flows in 4 different directions, with 2 speeds ($\leq$ 0.51 m/s), 3 densities (up to 1.5× of water), and 5 viscosities (up to 7,000× of water). A total of 120 conditions.
- (**P**) Impulse or random perturbations applied to the body, including short and strong forces, and long and light forces, varying in directions and magnitudes, providing 91 conditions.
- (**B**) 3 body geometries where we attach fins of 10cm, 15cm, and 20cm long to the feet, and 8 body morphologies by removing body parts. A total of 11 conditions.
- (**S**) 4 swimming styles: front-crawl, backstroke, breaststroke, and butterfly.

Theoretically, there are a total of 14,414,400 combinations, of which we choose 16,632 (see Appendix A). We report the results in Table 2 across all four swimming styles across **T**×**E**. Overall, SWIM achieves $\geq$ 98% SRV and $\leq$ 0.4 PS, showing that it can successfully exploit meaningful policies within valid physical spaces and motions, and accomplish tasks across various unseen conditions that are drastically different from that of training. This demonstrates robust task completion beyond the training distribution. Note that we have a strict **SR** criterion (detailed in Figure 5). Despite **SR** can be as low as 27.5%, the **Prog** shows SWIM has reasonably accomplished all tasks.

In goal-reaching, Figure 5 shows that SWIM generalizes up to $\theta_{lat} = 30°$ while maintaining a position error below 0.2 m. Beyond $\theta_{lat} = 30°$ in both tasks, flow speed larger than $0.51m/s$, and extreme fluid densities ($1500kg/m^3$), SWIM starts to fail more often. However, we argue these combinations are challenging for humans too, *e.g.* $0.51m/s$ is a dangerous flow for swimming already (NFPA, 2004), highlighting the difficulty of these unseen conditions in practice.

We show one example of trajectory-following in Figure 6, which shows SWIM can follow complex trajectories in unseen conditions, even only trained on a simple goal-reaching task. More quantitative and visual results in Appendix B.1 further confirm that SWIM has a strong generalization and produces natural-looking motions (see videos) across a wide range of unseen conditions.

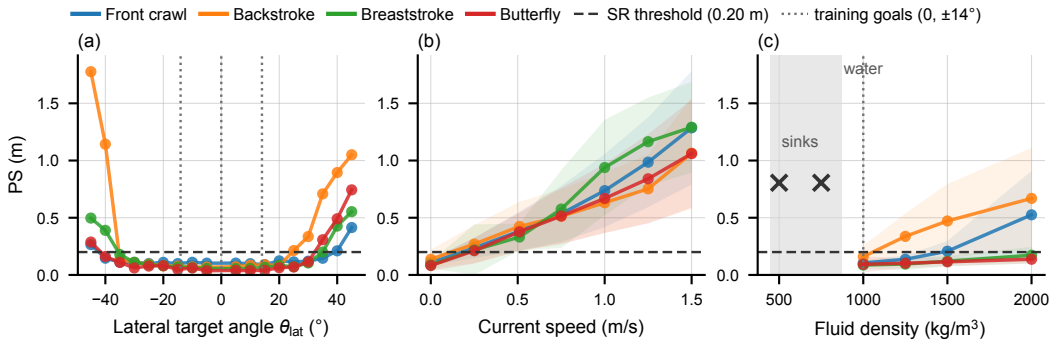

Figure 5: Position errors in different conditions. **(a)** Goal-reaching in the 5 m pool against the lateral target angle θ_{lat}; the dotted vertical lines are the three training goal angles (0 and $\pm 14°$). **(b)** Against the flow speed and averaged over four flow directions. **(c)** Against the fluid density; the dotted line is water, the training fluid. The character sinks in the shaded range (sinks) and hence no position error reported. In all figures, the horizontal dashed line is the SR threshold (PS ≤ 0.20 m).

4.3 Comparisons

As there are no baseline methods for direct comparison, we adapt several baseline methods in imitation learning for human motions: DeepMimic (Peng et al., 2018; Peng, 2025), AMP (Peng et al., 2021; Peng, 2025) and ADD (Zhang et al., 2025). We also use a simple PD controller to follow the reference motion (data replay) as a baseline. This comparison shows SWIM is genuinely the first RL method capable for synthesizing stable, controllable and natural swimming motions. Furthermore, we compare SWIM against three surrogate models in the transition function realization: a world model in CFC (Dou et al., 2025) based on Physics-inspired Neural Networks (PINNs), MuJoCo's inertial fluid model (Todorov et al., 2012; Vaxenburg et al., 2025), and OceanSim (Song et al., 2025a) designed for underwater robots. In addition, we use a high-resolution SPH simulator as another baseline which estimates water-body forces more accurately than SWIM. Overall, these baselines show that SWIM strikes a balance between accuracy and speed for effective RL learning. More details are provided in Appendix B.2.2.

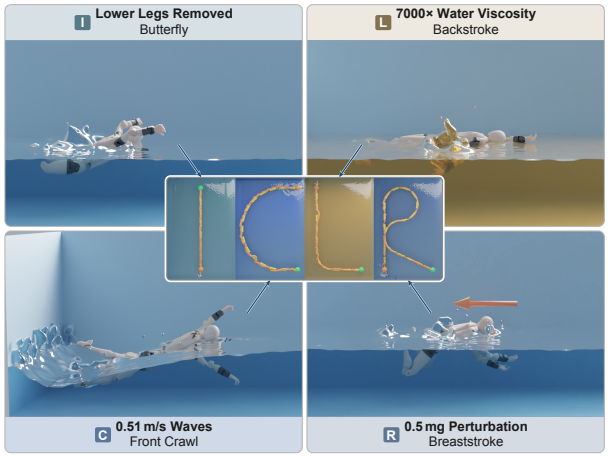

Figure 6: 'ICLR' trajectory-following under challenging conditions. Orange curves are from synthesized motions.

Results on **T**×**E** conditions averaged over all swimming styles are shown in Table 3. First, SWIM achieves the highest on key task-related metrics such as **SR** and **Prog** by large margin. Although AMP and DeepMimic occasionally achieve better results on task metrics, their motions are unnatural, with the arms moving in bird-like flapping motions (see videos). Overall, these results show that existing methods cannot be easily adapted to the SWIM task.

Also, Table 3 bottom shows that both surrogate models and highly accurate simulators achieve low SRs (below 10%), with varying speeds in generating samples for RL via simulation (samples/s). SWIM is neither the fastest nor the slowest, but is the best balanced one. Surrogate models are overly inaccurate for RL learning, *e.g.* MuJoCo not modeling dynamic water-air boundaries, while high-res simulators (High-resolution SPH) is too slow for RL training in practice.

Finally, we further compare our GPU-based SPH simulator with the original CPU-based SPlisHSPlasH (Bender et al., 2017). On the same machine, our simulator can run 18 environments in parallel and achieves $1{,}023$ samples/s, compared with merely 65 samples/s using 24 core CPU AMD Ryzen Threadripper 9960X CPU with SPlisHSPlasH (more details in Appendix D). This corre-

Table 3: **Comparison after training on 10M samples.** Top: Comparison with imitation learning methods; Bottom: Comparison with surrogate models. † trained for the same amount of time as SWIM or otherwise too slow if trained on 10M samples. The top-3 is labeled as 1st / 2nd / 3rd.

Method	SRV↑ (%)	SR↑ (%)	PS↓ (m)	LD↓ (m)	Prog→ 1	Roll↓ (°)	Vel↓ (rad/s)	Accel↓ (rad/s^2)	Samples/s↑
SWIM (ours)	$99.7_{\pm 0.2}$	$63.9_{\pm 4.7}$	$0.211_{\pm 0.024}$	$0.101_{\pm 0.024}$	$0.857_{\pm 0.015}$	$0.94_{\pm 0.54}$	$1.31_{\pm 0.24}$	$64.84_{\pm 14.59}$	1,023
AMP	$97.5_{\pm 2.1}$	$36.8_{\pm 44.6}$	$0.961_{\pm 0.801}$	$0.089_{\pm 0.032}$	$0.413_{\pm 0.499}$	$0.45_{\pm 0.46}$	$2.21_{\pm 0.76}$	$128.67_{\pm 58.98}$	940
DeepMimic	$98.7_{\pm 2.3}$	$12.7_{\pm 25.4}$	$1.236_{\pm 0.464}$	$0.099_{\pm 0.023}$	$0.219_{\pm 0.276}$	$3.35_{\pm 4.04}$	$2.76_{\pm 0.85}$	$323.75_{\pm 153.02}$	903
ADD	$98.4_{\pm 2.5}$	$0.0_{\pm 0.0}$	$1.597_{\pm 0.138}$	$0.122_{\pm 0.054}$	$0.028_{\pm 0.081}$	$0.46_{\pm 0.63}$	$1.65_{\pm 0.21}$	$107.37_{\pm 27.46}$	899
Data Replay	$100.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$0.900_{\pm 0.278}$	$0.264_{\pm 0.131}$	$0.403_{\pm 0.172}$	$3.36_{\pm 4.98}$	$0.73_{\pm 0.21}$	$35.98_{\pm 11.21}$	1,160
MuJoCo	$58.4_{\pm 40.8}$	$1.8_{\pm 3.5}$	$0.808_{\pm 0.137}$	$0.359_{\pm 0.182}$	$0.644_{\pm 0.184}$	$25.12_{\pm 23.24}$	$1.59_{\pm 0.16}$	$81.42_{\pm 17.15}$	2,194
CFC†	$87.7_{\pm 16.9}$	$0.0_{\pm 0.0}$	$1.133_{\pm 0.189}$	$0.205_{\pm 0.099}$	$0.267_{\pm 0.117}$	$10.61_{\pm 9.76}$	$0.90_{\pm 0.20}$	$53.03_{\pm 17.37}$	55
OceanSim	$68.2_{\pm 43.8}$	$0.0_{\pm 0.0}$	$1.237_{\pm 0.266}$	$0.328_{\pm 0.136}$	$0.251_{\pm 0.129}$	$15.68_{\pm 23.35}$	$1.20_{\pm 0.23}$	$67.30_{\pm 18.62}$	2,657
High-res SPH†	$74.6_{\pm 18.1}$	$0.0_{\pm 0.0}$	$1.266_{\pm 0.157}$	$0.232_{\pm 0.110}$	$0.204_{\pm 0.086}$	$18.02_{\pm 14.83}$	$1.11_{\pm 0.18}$	$61.76_{\pm 20.02}$	5

Table 4: **Environment state ablation.** The top-3 per column is labeled as 1st / 2nd / 3rd.

Environment state	SRV↑ (%)	SR↑ (%)	PS↓ (m)	LD↓ (m)	Prog→ 1	Roll↓ (°)	Vel↓ (rad/s)	Accel↓ (rad/s^2)
SWIM	$99.7_{\pm 0.2}$	$63.9_{\pm 4.7}$	$0.211_{\pm 0.024}$	$0.101_{\pm 0.024}$	$0.857_{\pm 0.015}$	$0.94_{\pm 0.54}$	$1.31_{\pm 0.24}$	$64.84_{\pm 14.59}$
Whole-body token	$98.2_{\pm 3.4}$	$63.8_{\pm 6.1}$	$0.210_{\pm 0.055}$	$0.104_{\pm 0.050}$	$0.874_{\pm 0.015}$	$1.64_{\pm 2.01}$	$1.34_{\pm 0.20}$	$66.35_{\pm 13.17}$
Per-body token	$96.4_{\pm 6.8}$	$68.5_{\pm 4.6}$	$0.193_{\pm 0.022}$	$0.101_{\pm 0.025}$	$0.871_{\pm 0.015}$	$2.46_{\pm 3.28}$	$1.34_{\pm 0.25}$	$67.23_{\pm 15.13}$
Body-surface-patch token	$98.2_{\pm 3.4}$	$56.3_{\pm 7.9}$	$0.232_{\pm 0.030}$	$0.121_{\pm 0.019}$	$0.850_{\pm 0.024}$	$1.52_{\pm 1.62}$	$1.36_{\pm 0.26}$	$67.71_{\pm 16.66}$
Whole-body raw	$99.6_{\pm 0.7}$	$69.8_{\pm 3.6}$	$0.177_{\pm 0.018}$	$0.087_{\pm 0.015}$	$0.896_{\pm 0.013}$	$0.74_{\pm 0.76}$	$1.36_{\pm 0.20}$	$67.82_{\pm 12.60}$
Per-body raw	$95.6_{\pm 8.5}$	$66.8_{\pm 6.6}$	$0.207_{\pm 0.048}$	$0.112_{\pm 0.034}$	$0.876_{\pm 0.015}$	$2.93_{\pm 4.69}$	$1.31_{\pm 0.20}$	$71.95_{\pm 16.12}$
Body-surface-patch raw	$97.7_{\pm 4.1}$	$59.2_{\pm 9.7}$	$0.228_{\pm 0.065}$	$0.114_{\pm 0.054}$	$0.847_{\pm 0.010}$	$1.79_{\pm 2.30}$	$1.28_{\pm 0.25}$	$73.30_{\pm 17.01}$
High-dim state	$86.6_{\pm 26.9}$	$0.0_{\pm 0.0}$	$1.067_{\pm 0.173}$	$0.206_{\pm 0.044}$	$0.308_{\pm 0.110}$	$4.77_{\pm 7.11}$	$0.87_{\pm 0.22}$	$43.63_{\pm 7.60}$

sponds to a higher sampling throughput of more than $15\times$. In general, these results demonstrate that SWIM achieves a favorable balance between physical accuracy and training efficiency.

4.4 Ablation Study

We test alternative environment representations in two dimensions: *spatial resolution*, the granularity at which water-body forces are represented: whole-body, per-body, or body-surface patches (52 patches in total), and *feature representation*, which specifies whether the water-body forces are represented as raw forces or tokens. Additionally, we test a feature combination of whole-body raw forces + per-body token (SWIM). Finally, we test densely sampled points on the body surface over time, forming a high-dimensional, unstructured state (High-dim state). We show one ablation study in Table 4 under $\mathbf{T}\times\mathbf{E}$ conditions averaged over all swimming styles here and additional ablation study are provided in Appendix B.2. The detailed settings are provided in Appendix D.

Overall, most high-dimensional representations (*e.g.* Body-surface-patch raw/token, High-dim state) and unstructured representations (*e.g.* Whole-body token, Per-body raw) perform badly. Tokenization is useful particularly at per-body level, *e.g.* Per-body token. Whole-body raw and Per-body token are overall winners but the former produces less natural motions than the latter. Therefore, we choose Whole-body raw + Per-body token for SWIM, which combines an overall wrench plus quantized per-body water-body forces as a balance between tasks and motion quality.

5 Conclusion, Limitation, and Future Work

We proposed SWIM, the first RL-based method for physically-based character swimming. By combining a structured low-dimensional environment representation and an efficient simulator, SWIM learns natural, controllable, and generalizable swimming control from a single motion. Extensive experiments demonstrate robust performance across tasks, environments, body geometries and morphologies, and swimming styles under a wide range of unseen conditions.

One limitation is that SWIM relies on motion data, the quality of which should be sufficient to guide sampling in RL training, as we only use one reference motion. Also, the learned policy has limited generalization in significant changes of setting, *e.g.* strong waves, large fluid densities, strong perturbations, and large changes in body morphology, *etc*. In future, we will incorporate lower quality motion data such as those reconstructed from videos to broaden the applicability of SWIM. We will also explore curriculum learning to gradually train the character from simple tasks, a standard body, and quiet water, to progressively harder settings to improve the generalizability.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks: discussing and providing feedback on experiment designs and experimental setups; providing feedback on research methodology and experiments; implementing methods using vibe coding;

We have not used generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, support qualitative and thematic data analysis, or interpret results. Assistance with translation is not applicable to this work.

Additionally, we used generative AI tools to create or modify scientific figures or images. We reviewed all AI-assisted work. AI-generated code for the proposed method, baselines, and synthetic data generation was verified by the authors. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies physically simulated swimming and involves no human subjects, no personally identifying data, and no user study. The reference motions are published biomechanical measurements obtained from prior work, and are used only to drive a simulated character. We see no direct pathway from this work to harmful application beyond those already inherent to physics-based character animation. The compute cost of the training campaign is reported.

REPRODUCIBILITY STATEMENT

We will share our data and source code upon acceptance. The simulator configuration, the observation layout, the reward terms, the network architecture and the full set of training hyperparameters are specified in the appendix. The evaluation protocol, including the pool geometries, the goal and trajectory families, and the definition of every reported metric, is given in the appendix as well, so that the reported numbers can be reproduced from the released code.

REFERENCES

Nadir Akinci, Markus Ihmsen, Gizem Akinci, Barbara Solenthaler, and Matthias Teschner. Versatile rigid-fluid coupling for incompressible SPH. *ACM Transactions on Graphics*, 31(4):1–8, 2012. ISSN 0730-0301, 1557-7368. doi: 10.1145/2185520.2185558. URL https://dl.acm.org/doi/10.1145/2185520.2185558.

Jan Bender and Dan Koschier. Divergence-free smoothed particle hydrodynamics. In *Proceedings of the 14th ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pp. 147–155, Los Angeles California, 2015. ACM. ISBN 978-1-4503-3496-9. doi: 10.1145/2786784.2786796. URL https://dl.acm.org/doi/10.1145/2786784.2786796.

Jan Bender et al. SPlisHSPlasH Library, 2017. URL https://github.com/InteractiveComputerGraphics/SPlisHSPlasH.

Kevin Bergamin, Simon Clavet, Daniel Holden, and James Richard Forbes. DReCon: data-driven responsive control of physics-based characters. *ACM Transactions on Graphics (TOG)*, 38(6): 1–11, 2019.

Gang Chen, Yuwang Lu, Xin Yang, and Huosheng Hu. Reinforcement learning control for the swimming motions of a beaver-like, single-legged robot based on biological inspiration. *Robotics and Autonomous Systems*, 154:104116, 2022. ISSN 09218890. doi: 10.1016/j.robot.2022.104116. URL https://linkinghub.elsevier.com/retrieve/pii/S0921889022000653.

Shuguang Chu, Zebin Huang, Yutong Li, Mingwei Lin, Ignacio Carlucho, Yvan R. Petillot, and Canjun Yang. MarineGym: A high-performance reinforcement learning platform for underwater robotics, 2025. URL http://arxiv.org/abs/2503.09203.

Francesco Crenna, Giovanni Battista Rossi, and Marta Berardengo. Filtering biomechanical signals in movement analysis. *Sensors*, 21(13):4580, 2021. ISSN 1424-8220. doi: 10.3390/s21134580. URL https://www.mdpi.com/1424-8220/21/13/4580.

Zhiyang Dou, Chen Peng, Xinyu Lu, Xiaohan Ye, Lixing Fang, Yuan Liu, Wenping Wang, Chuang Gan, Lingjie Liu, and Taku Komura. CFC: Simulating character-fluid coupling using a two-level world model. *ACM Transactions on Graphics*, 44(6):1–17, 2025. ISSN 0730-0301, 1557-7368. doi: 10.1145/3763318. URL https://dl.acm.org/doi/10.1145/3763318.

Tao Du, Josie Hughes, Sebastien Wah, Wojciech Matusik, and Daniela Rus. Underwater soft robot modeling and control with differentiable simulation. *IEEE Robotics and Automation Letters*, 6 (3):4994–5001, 2021. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2021.3070305. URL https://ieeexplore.ieee.org/document/9392257/.

Aaron English, Jose M Domínguez, Renato Vacondio, Alejandro JC Crespo, Peter K Stansby, SJ Lind, Luca Chiapponi, and M Gómez-Gesteira. Modified dynamic boundary conditions (mdbc) for general-purpose smoothed particle hydrodynamics (sph): application to tank sloshing, dam break and fish pass problems. *Computational Particle Mechanics*, 9(5):1–15, 2022.

Yu Fang, Ziyin Qu, Minchen Li, Xinxin Zhang, Yixin Zhu, Mridul Aanjaneya, and Chenfanfu Jiang. IQ-MPM: an interface quadrature material point method for non-sticky strongly two-way coupled nonlinear solids and fluids. *ACM Transactions on Graphics*, 39(4), 2020. ISSN 0730-0301, 1557-7368. doi: 10.1145/3386569.3392438. URL https://dl.acm.org/doi/10.1145/3386569.3392438.

Christoph Gissler, Andreas Peer, Stefan Band, Jan Bender, and Matthias Teschner. Interlinked SPH pressure solvers for strong fluid-rigid coupling. *ACM Transactions on Graphics*, 38(1):1–13, 2019. ISSN 0730-0301, 1557-7368. doi: 10.1145/3284980. URL https://dl.acm.org/doi/10.1145/3284980.

Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal Mehta, and João G.M. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL http://jmlr.org/papers/v23/21-1342.html.

Pranav Jain, Ziyin Qu, Peter Yichen Chen, and Oded Stein. Neural monte carlo fluid simulation. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pp. 1–11, Denver CO USA, 2024. ACM. doi: 10.1145/3641519.3657438. URL https://dl.acm.org/doi/10.1145/3641519.3657438.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. URL http://arxiv.org/abs/1412.6980. arXiv:1412.6980 [cs.LG].

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. Smoothed particle hydrodynamics techniques for the physics based simulation of fluids and solids. *Eurographics 2019 - Tutorials*, pp. 41 pages, 2019. ISSN 1017-4656. doi: 10.2312/egt.20191035. URL http://arxiv.org/abs/2009.06944.

N. Kwatra, C. Wojtan, M. Carlson, I.A. Essa, P.J. Mucha, and G. Turk. Fluid simulation with articulated bodies. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):70–80, 2010. ISSN 1077-2626. doi: 10.1109/TVCG.2009.66. URL http://ieeexplore.ieee.org/document/5089323/.

LABIOMEP and Qualisys. Swimming techniques optical motion capture dataset. Qualisys File Library, project *Swimming Techniques*, 2025. URL https://qfl.qualisys.com/#!/project/swimming-techniques. Porto Biomechanics Laboratory, University of Porto; multi-camera synchronized underwater and poolside motion capture; accessed April 2025.

M Lentine, J T Gretarsson, C Schroeder, A Robinson-Mosher, and R Fedkiw. Creature control in a fluid environment. *IEEE Transactions on Visualization and Computer Graphics*, 17(5):682–693, 2011. ISSN 1077-2626. doi: 10.1109/TVCG.2010.108. URL http://ieeexplore.ieee.org/document/5557867/.

Zhehao Li, Qingyu Xu, Xiaohan Ye, Bo Ren, and Ligang Liu. DiffFR: Differentiable SPH-based fluid-rigid coupling for rigid body control. *ACM Transactions on Graphics*, 42(6):1–17, 2023. ISSN 0730-0301, 1557-7368. doi: 10.1145/3618318. URL https://dl.acm.org/doi/10.1145/3618318.

Libin Liu and Jessica Hodgins. Learning to schedule control fragments for physics-based characters using deep Q-learning. *ACM Transactions on Graphics*, 36(3):1–14, 2017. ISSN 0730-0301, 1557-7368. doi: 10.1145/3083723. URL https://dl.acm.org/doi/10.1145/3083723.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts, May 2017. URL http://arxiv.org/abs/1608.03983. arXiv:1608.03983 [cs.LG].

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, January 2019. URL http://arxiv.org/abs/1711.05101. arXiv:1711.05101 [cs.LG].

Miles Macklin. Warp: A high-performance python framework for GPU simulation and graphics, 3 2022. URL https://github.com/NVIDIA/warp. NVIDIA GPU Technology Conference (GTC).

Nick McGreivy and Ammar Hakim. Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations. *Nature machine intelligence*, 6 (10):1256–1269, 2024.

Siddharth Mysore, Bassel Mabsout, Kate Saenko, and Renato Mancuso. How to train your quadrotor: A framework for consistently smooth and responsive flight control via reinforcement learning. *ACM Transactions on Cyber-Physical Systems (TCPS)*, 5(4):1–24, 2021.

NFPA. *NFPA 1670 Standard on Operations and Training for Technical Search and Rescue Incidents*. National Fire Protection Association, 2004.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318. Pmlr, 2013.

DR Pendergast, PE Di Prampero, AB Craig Jr, DR Wilson, and DW Rennie. Quantitative analysis of the front crawl in men and women. *Journal of Applied Physiology*, 43(3):475–479, 1977.

Xue Bin Peng. MimicKit: A reinforcement learning framework for motion imitation and control. *arXiv preprint arXiv:2510.13794*, 2025.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van De Panne. DeepMimic: example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics*, 37(4):1–14, 2018. ISSN 0730-0301, 1557-7368. doi: 10.1145/3197517.3201311. URL https://dl.acm.org/doi/10.1145/3197517.3201311.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics*, 40 (4), 2021. ISSN 0730-0301, 1557-7368. doi: 10.1145/3450626.3459670.

Stelios G Psycharakis and Ross H Sanders. Body roll in swimming: A review. *Journal of Sports Sciences*, 28(3):229–236, 2010.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045. URL https://linkinghub.elsevier.com/retrieve/pii/S0021999118307125. Publisher: Elsevier BV.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pp. 8459–8468. PMLR, 2020.

Daiki Satoi, Mikihiro Hagiwara, Akira Uemoto, Hisanao Nakadai, and Junichi Hoshino. Unified motion planner for fishes with various swimming styles. *ACM Transactions on Graphics (TOG)*, 35(4):1–15, 2016.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL http://arxiv.org/abs/1707.06347.

Yi Shi, Yifeng Jiang, Chen Tessler, and Xue Bin Peng. GPC: Large-scale generative pretraining for transferable motor control. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pp. 1–11. ACM, 2026. ISBN 979-8-4007-2554-8. doi: 10.1145/3799902.3811038. URL https://dl.acm.org/doi/10.1145/3799902.3811038.

Weiguang Si, Sung-Hee Lee, Eftychios Sifakis, and Demetri Terzopoulos. Realistic biomechanical simulation and control of human swimming. *ACM Transactions on Graphics*, 34(1):1–15, 2014. ISSN 0730-0301, 1557-7368. doi: 10.1145/2626346. URL https://dl.acm.org/doi/10.1145/2626346.

Jingyu Song, Haoyu Ma, Onur Bagoren, Advaith V. Sethuraman, Yiting Zhang, and Katherine A. Skinner. OceanSim: A GPU-accelerated underwater robot perception simulation framework, 2025a. URL http://arxiv.org/abs/2503.01074.

Wenbin Song, Heng Zhang, Yang Wang, and Xiaopei Liu. Creating fluid-interactive virtual agents by an efficient simulator with local-domain control. *ACM Transactions on Graphics (SIGGRAPH 2025)*, 2025b. Accepted for publication. A video clip was selected for the Technical Papers Trailer.

Tetsuya Takahashi and Christopher Batty. Monolith: a monolithic pressure-viscosity-contact solver for strong two-way rigid-rigid rigid-fluid coupling. *ACM Transactions on Graphics*, 39(6):1–16, 2020. ISSN 0730-0301, 1557-7368. doi: 10.1145/3414685.3417798. URL https://dl.acm.org/doi/10.1145/3414685.3417798.

Tetsuya Takahashi and Christopher Batty. ElastoMonolith: A monolithic optimization-based liquid solver for contact-aware elastic-solid coupling. *ACM Transactions on Graphics (TOG)*, 41(6):1–19, 2022.

Jie Tan, Yuting Gu, Greg Turk, and C. Karen Liu. Articulated swimming creatures. In *ACM SIGGRAPH 2011 papers*, pp. 1–12, Vancouver British Columbia Canada, 2011. ACM. ISBN 978-1-4503-0943-1. doi: 10.1145/1964921.1964953. URL https://dl.acm.org/doi/10.1145/1964921.1964953.

Chen Tessler, Yunrong Guo, Ofir Nabati, Gal Chechik, and Xue Bin Peng. MaskedMimic: Unified physics-based character control through masked motion inpainting. *ACM Transactions on Graphics*, 43(6):1–21, 2024. ISSN 0730-0301, 1557-7368. doi: 10.1145/3687951. URL https://dl.acm.org/doi/10.1145/3687951.

Nils Thuerey, Philipp Holl, Maximilian Mueller, Patrick Schnell, Felix Trost, and Kiwon Um. Physics-based deep learning. *arXiv preprint arXiv:2109.05237*, 2021.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Roman Vaxenburg, Igor Siwanowicz, Josh Merel, Alice A. Robie, Carmen Morrow, Guido Novati, Zinovia Stefanidi, Gert-Jan Both, Gwyneth M. Card, Michael B. Reiser, Matthew M. Botvinick, Kristin M. Branson, Yuval Tassa, and Srinivas C. Turaga. Whole-body physics simulation of fruit fly locomotion. *Nature*, 2025. ISSN 1476-4687. doi: 10.1038/s41586-025-09029-4. URL https://doi.org/10.1038/s41586-025-09029-4.

Kevin Wang, Ishaan Javali, Michał Bortkiewicz, Tomasz Trzcinski, and Benjamin Eysenbach. 1000 layer networks for self-supervised rl: Scaling depth can enable new goal-reaching capabilities. *Advances in Neural Information Processing Systems*, 38:157643–157670, 2026.

Xiaokun Wang, Yanrui Xu, Sinuo Liu, Bo Ren, Jirí Kosinka, Alexandru C Telea, Jiamin Wang, Chongming Song, Jian Chang, Chenfeng Li, et al. Physics-based fluid simulation in computer graphics: Survey, research trends, and challenges. *Computational Visual Media*, 10(5):803–858, 2024.

Yinhuai Wang, Qihan Zhao, Runyi Yu, Hok Wai Tsui, Ailing Zeng, Jing Lin, Zhengyi Luo, Jiwen Yu, Xiu Li, Qifeng Chen, Jian Zhang, Lei Zhang, and Ping Tan. SkillMimic: Learning basketball interaction skills from demonstrations, 2025. URL http://arxiv.org/abs/2408.15270.

Marcel Weiler, Dan Koschier, Magnus Brand, and Jan Bender. A Physically Consistent Implicit Viscosity Solver for SPH Fluids. *Computer Graphics Forum*, 37(2):145–155, May 2018. ISSN 0167-7055, 1467-8659. doi: 10.1111/cgf.13349. URL https://onlinelibrary.wiley.com/doi/10.1111/cgf.13349.

Michael Xu, Yi Shi, KangKang Yin, and Xue Bin Peng. PARC: Physics-based augmentation with reinforcement learning for character controllers. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pp. 1–11, Vancouver BC Canada, 2025. ACM. ISBN 979-8-4007-1540-2. doi: 10.1145/3721238.3730616. URL https://dl.acm.org/doi/10.1145/3721238.3730616.

Pei Xu, Yufei Ye, Shuchun Sun, Yu Ding, Elizabeth Schumann, and C. Karen Liu. MUSIC: Learning muscle-driven dexterous hand control. *ACM Transactions on Graphics*, 45(4), 2026. doi: 10.1145/3811402.

Po-Feng Yang, Joe Laszlo, and Karan Singh. Layered dynamic control for interactive character swimming. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation - SCA '04*, pp. 39, Grenoble, France, 2004. ACM Press. ISBN 978-3-905673-14-2. doi: 10.1145/1028523.1028529. URL http://portal.acm.org/citation.cfm?doid=1028523.1028529. ISSN: 17275288.

Zeshi Yang, Kangkang Yin, and Libin Liu. Learning to use chopsticks in diverse gripping styles. *ACM Transactions on Graphics*, 41(4):1–17, 2022. ISSN 0730-0301, 1557-7368. doi: 10.1145/3528223.3530057. URL https://dl.acm.org/doi/10.1145/3528223.3530057.

Heyuan Yao, Zhenhua Song, Baoquan Chen, and Libin Liu. ControlVAE: Model-based learning of generative controllers for physics-based characters. *ACM Transactions on Graphics*, 41(6):1–16, 2022. ISSN 0730-0301, 1557-7368. doi: 10.1145/3550454.3555434. URL http://arxiv.org/abs/2210.06063.

Heyuan Yao, Zhenhua Song, Yuyang Zhou, Tenglong Ao, Baoquan Chen, and Libin Liu. MoConVQ: Unified physics-based motion control via scalable discrete representations, 2023. URL http://arxiv.org/abs/2310.10198.

Ye Yuan and Kris Kitani. Residual force control for agile human behavior imitation and extended motion synthesis. *Advances in Neural Information Processing Systems*, 33:21763–21774, 2020.

Haotian Zhang, Ye Yuan, Viktor Makoviychuk, Yunrong Guo, Sanja Fidler, Xue Bin Peng, and Kayvon Fatahalian. Learning physically simulated tennis skills from broadcast videos. *ACM Transactions on Graphics*, 42(4):1–14, 2023. ISSN 0730-0301, 1557-7368. doi: 10.1145/3592408. URL https://dl.acm.org/doi/10.1145/3592408.

Ziyu Zhang, Sergey Bashkirov, Dun Yang, Yi Shi, Michael Taylor, and Xue Bin Peng. Physics-based motion imitation with adversarial differential discriminators. In *SIGGRAPH Asia 2025 Conference Papers (SIGGRAPH Asia '25 Conference Papers)*, 2025.

Appendix

CONTENTS

A Detailed Experimental Setup **19**

A.1 Task Generalization Settings 19

A.2 Environments Generalization Settings 21

A.3 Perturbation Generalization Settings 21

A.4 Body Generalization Settings 21

A.5 Metric Definitions and Justifications 22

A.6 Reference Motion Data 24

A.7 Training vs Generalization setup 24

B Additional Experimental Results **25**

B.1 Detailed and Additional Generalization 25

B.2 Comparison and Ablation Study 40

B.3 Additional Ablation Study for the Dual Directed-Branch GCN VQ-VAE tokenizer . 44

C Full Details of Methodology **49**

C.1 State Representation Details 49

C.2 Policy and Action Space 51

C.3 Coupled Body–Fluid Simulation 52

C.4 Reward Detail 53

C.5 Learning Algorithm 54

D Implementation Details **54**

D.1 GPU-based SPH simulator 54

D.2 Humanoid Character 56

D.3 Other Training and Evaluation Configurations 57

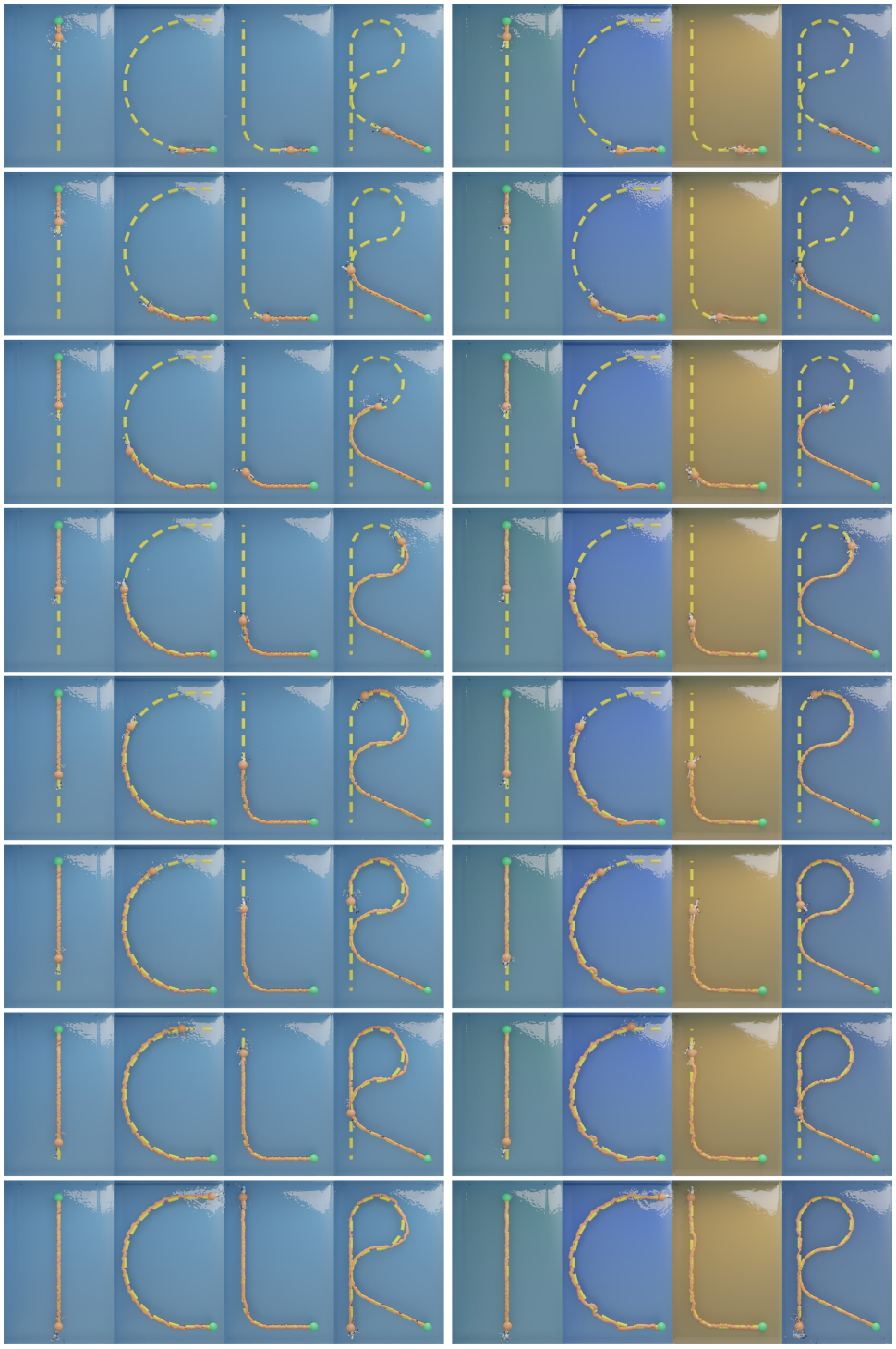

Figure 7: **Writing ICLR.** Four trajectory-following episodes, one per letter: I (butterfly), C (front crawl), L (backstroke), and R (breaststroke). **Left:** all four letters in still water. **Right:** I with both lower leg removed, C in a 0.51 m/s wave along the pool, L in honey, and R in $0.5\,mg$ perturbation every 6 s.



Figure 8: **One stroke cycle, four strokes.** Side view on the goal-reaching task in still water. Each stroke takes two rows with six key poses from one cycle, read left to right and top to bottom. The cycle length differs by stroke, from $1.06\,\mathrm{s}$ for butterfly to $1.58\,\mathrm{s}$ for front crawl.

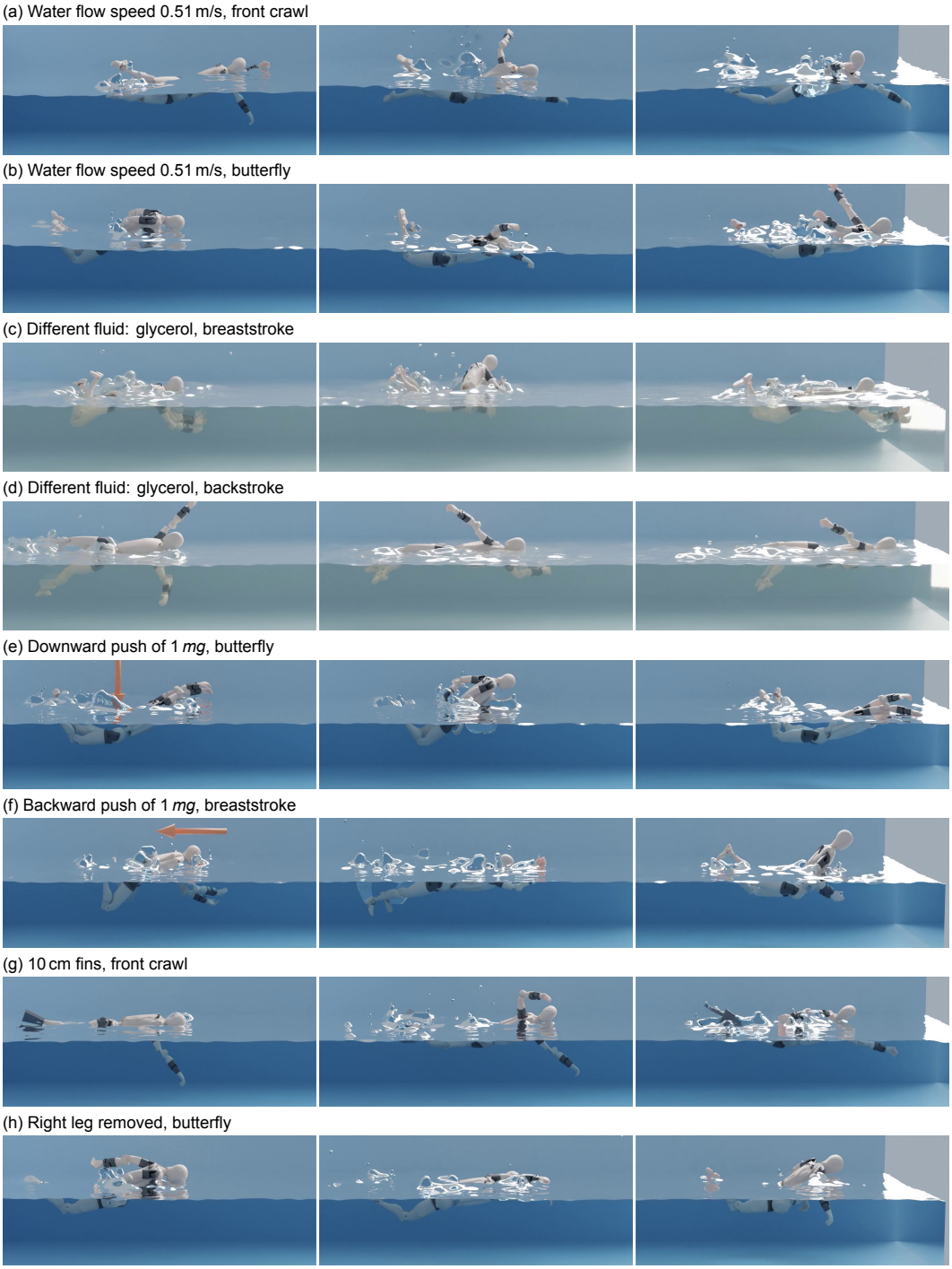

Figure 9: **Unseen conditions, side view.** Each row shows one SWIM experiment, with the view following the character. Frames are shown at approximately 2, 5, and 8 s, except in (e) and (f), where they are captured during the push at approximately 4.5 s and then 1 and 3 s later. The orange arrow marks the push, with a force of one body weight ($1\,mg$) applied for 0.1 s. The policy maintains control during the push and recovers afterward. Every episode shown reaches its goal.

Table 5: **Number of experiments.** One experiment is one policy swimming one task under one condition. Rows naming several methods or arms multiply by that count.

	Conditions	Tasks	Strokes	Seeds	Experiments	Subtotal
Strong generalization (Section 4.2)						
$\mathbf{T} \times \mathbf{E1}$ (flow)	8	30	4	1	960	16,632
$\mathbf{T} \times \mathbf{E2}$ (fluid)	8	30	4	1	960	
$\mathbf{T} \times \mathbf{P}$ (perturbation)	91	30	4	1	10,920	
$\mathbf{T} \times \mathbf{B}$ (body)	11	30	4	1	1,320	
$\mathbf{T}$ in the $4\times$ pool	1	30	4	1	120	
Goal-reaching angle sweep, $\pm5^\circ$ to $\pm45^\circ$	1	18	4	1	72	
$\mathbf{T}\times$ additional flow speeds	16	30	4	1	1,920	
$\mathbf{T}\times$ additional densities	3	30	4	1	360	
Baselines (Table 3), on $\mathbf{T} \times \mathbf{E1}$						
Imitation baselines (MimicKit-AMP, MimicKit-DeepMimic, ADD)	8	30	4	3	8,640	20,160
Transition-function baselines (MuJoCo, CFC, OceanSim, high-res SPH)	8	30	4	3	11,520	
Ablation study (Tables 4 and 14), on $\mathbf{T} \times \mathbf{E1}$						
Environment state (8 arms)	8	30	4	3	23,040	46,080
Reward, goal state and action space (8 arms)	8	30	4	3	23,040	
Total						**82,872**

Table 6: **Number of trained policies.** Each method is trained from scratch with three seeds for each of the four strokes. The generalization experiments in Table 5 reuse the best seed for each stroke without any additional training.

	Methods	Strokes	Seeds	Policies
Environment state	8	4	3	96
Reward, goal state and action space	8	4	3	96
Imitation baselines (MimicKit-AMP, MimicKit-DeepMimic, ADD)	3	4	3	36
Transition-function baselines (MuJoCo, CFC, OceanSim, high-res SPH)	4	4	3	48
Total				**276**

A Detailed Experimental Setup

We provide the detailed training and generalization settings in the main paper. Table 5 first summarizes the number of experiments underlying each part of the paper and the tasks and conditions they combine, and Table 6 gives the number of trained policies. The detailed conditions and tasks referred to in Table 5 will be explained in the rest of Appendix A.

A.1 Task Generalization Settings

Goal-reaching As mentioned in Section 4.1, we use 10 unseen goals parameterized by θ_{lat} including 4 unseen interpolation angles and 6 unseen extrapolation angles. For extrapolation, our policy extrapolation is limited to around $\theta_{\text{max}} = 30.0^\circ$, about $2\times$ the maximum training angle $\theta_{\text{train}} = 14.04^\circ$. Next, we also use two swimming pools, 3.0m and 6.0m long respectively, compared with the training pool which is only 1.5m long. Together, the goal positions are set based on the swimming distance and θ_{lat} explained below.

Interpolation: 4 points evenly spaced between 0° and $\pm\theta_{\text{train}}$:

$$\pm\tfrac{1}{3}\theta_{\text{train}} \approx \pm4.7^\circ, \qquad \pm\tfrac{2}{3}\theta_{\text{train}} \approx \pm9.4^\circ.$$

Extrapolation: 6 points evenly spaced between $\pm\theta_{\text{train}}$ and $\pm\theta_{\text{max}}$:

$$\pm\big(\theta_{\text{train}} + \tfrac{1}{3}(\theta_{\text{max}} - \theta_{\text{train}})\big) \approx \pm19.4^\circ, \pm\big(\theta_{\text{train}} + \tfrac{2}{3}(\theta_{\text{max}} - \theta_{\text{train}})\big) \approx \pm24.7^\circ, \pm\theta_{\text{max}} = \pm30.0^\circ.$$

Trajectory-Following Setup As mentioned in Section 4.1, we consider two types of turns: short, sharp turns and slow, long turns, represented by polyline and sinusoidal trajectories, respectively. We employ trajectory-following tasks across 20 trajectories, parameterized by a maximum angular steering limit $\theta_{\max} = 30.0^\circ$. Figure 10 shows the three types.

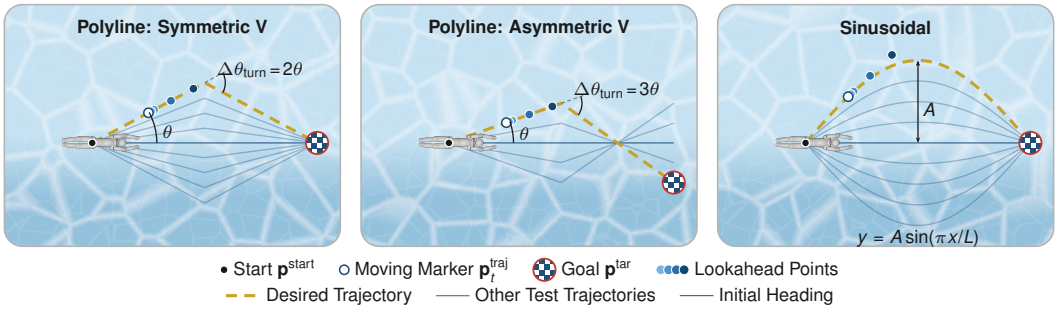

Figure 10: **Trajectory-following generalization setup.** The three trajectory types, drawn for the 3 m pool. Each part shows one group of trajectories controlled by a parameter: the symmetric V with $\theta = 15^\circ$ (8 curves), the asymmetric V with $\theta = 10^\circ$ (4 curves) and the sinusoid with $A = A_{\max} = 0.55$ m (8 curves). The character starts at $\mathbf{p}^{\text{start}}$ facing right. The moving marker $\mathbf{p}_t^{\text{traj}}$ travels along the desired trajectory at constant speed, and the four lookahead points are placed at $\delta t_i \in \{0.1, 0.2, 0.8, 1.6\}$ s ahead of $\mathbf{p}_t^{\text{traj}}$ along its current direction, so they do not need to be on the trajectory.

Polyline trajectories. We evaluate symmetric and asymmetric V-shaped polylines to test how well the policy responds to sudden changes in the target heading.

- **Symmetric V-shapes (8 curves):** Each trajectory consists of two linear segments. The character follows a heading of $+\theta$ during the first half and then changes to $-\theta$ during the second half, returning to the centerline. This creates an instantaneous turn of $\Delta\theta_{\text{turn}} = 2\theta$ at the midpoint. We use four uniformly spaced values for the heading angle:

$$\theta \in \left[\pm\frac{1}{8}\theta_{\max},\ \pm\frac{2}{8}\theta_{\max},\ \pm\frac{3}{8}\theta_{\max},\ \pm\frac{4}{8}\theta_{\max}\right] = [\pm 3.75^\circ,\ \pm 7.50^\circ,\ \pm 11.25^\circ,\ \pm 15.00^\circ], \tag{7}$$

  yielding vertex turning angles of $\Delta\theta_{\text{turn}} \in \{7.5^\circ, 15.0^\circ, 22.5^\circ, 30.0^\circ\}$.

- **Asymmetric V-shapes (4 curves):** The first segment goes from $(0, 0)$ to (middle, a) with a heading angle of $+\theta$. The second segment then goes from (middle, a) to $(2{*}\text{middle}, -a)$ with twice the slope, corresponding to a heading angle of -2θ. This creates a larger in-place turn of $\Delta\theta_{\text{turn}} = \theta - (-2\theta) = 3\theta$. This setup tests whether tracking error from the first segment affects the performance during the second segment. We use:

$$\theta \in \left[\pm\frac{1}{6}\theta_{\max},\ \pm\frac{2}{6}\theta_{\max}\right] = [\pm 5.0^\circ,\ \pm 10.0^\circ], \tag{8}$$

  corresponding to in-place turning angles of $\Delta\theta_{\text{turn}} \in \{15.0^\circ, 30.0^\circ\}$.

Sinusoidal trajectories (8 curves). To evaluate continuous heading changes, we use half-period sinusoidal curves defined by $y(x) = A\sin(\pi x/L)$, where L is the swimming distance. Each curve reaches a maximum lateral displacement of $d_{\text{lat}} = \pm A$, where as d_{lat} defined in Figure 4, at the midpoint and returns smoothly to the centerline at $x = L$. Unlike piecewise-linear paths, sinusoidal trajectories have continuously changing heading and curvature, with peak curvature $\kappa_{\max} = A(\pi/L)^2 \propto A$. We use four uniformly spaced amplitude levels:

$$A \in \left[\pm\frac{1}{4}A_{\max},\ \pm\frac{2}{4}A_{\max},\ \pm\frac{3}{4}A_{\max},\ \pm A_{\max}\right]. \tag{9}$$

These amplitudes correspond to peak tangent angles of $8.3^\circ, 16.1^\circ, 23.4^\circ, 30.0^\circ$. They cover interpolation and extrapolation cases relative to the training distribution, allowing us to test the limits of the learned controller.

A.2 Environments Generalization Settings

A robust policy should be able to swim in flowing water such as in the sea. We are also interested in evaluating how the policy behaves in fluids with physical properties different from those used during training. We therefore divide the unseen environments into two categories:

- (**E1: Flow**) Water flows in four initial flow directions: downstream ($0°$), diagonal ($45°$), crossflow ($90°$), and upstream ($180°$) with two flow speed 0.25m/s and 0.51m/s, resulting in 8 conditions.

  These directions perturb the character differently: downstream flow is along the swimming direction and cause unintended forward acceleration for swimming motions; upstream flow is the opposite to downstream motion, causing resistance; crossflow comes from a perpendicular direction to the swimming direction, causing constant lateral drift; diagonal flow combines perturbations of downstream and crossflow. In addition, because the pool boundaries, the flow field naturally oscillates and decays over time rather than remaining perfectly steady. Therefore, the direction labels here refer to the initial flow direction.

- (**E2: Fluid**) We employ 8 fluids with various densities and viscosities. We first use viscosity of water and vary the densities: $1000\,\mathrm{kg/m^3}$, $1250\,\mathrm{kg/m^3}$, $1500\,\mathrm{kg/m^3}$. In addition, we also use five real fluids: olive oil ($911\,\mathrm{kg/m^3}$, $8.4 \times 10^{-2}\,\mathrm{Pa \cdot s}$), seawater ($1025\,\mathrm{kg/m^3}$, $1.08 \times 10^{-3}\,\mathrm{Pa \cdot s}$), heavy water ($1107\,\mathrm{kg/m^3}$, $1.25 \times 10^{-3}\,\mathrm{Pa \cdot s}$), glycerol ($1261\,\mathrm{kg/m^3}$, $1.41\,\mathrm{Pa \cdot s}$), and honey ($1420\,\mathrm{kg/m^3}$, $10.0\,\mathrm{Pa \cdot s}$), for a total of 8 conditions.

Finally, we further evaluate the policy beyond the representative conditions in $\mathbf{E1}$ and $\mathbf{E2}$ by varying flow speed and fluid density. For $\mathbf{T} \times \mathbf{E1}$, we evaluate flow speeds of 0.25, 0.51, 0.75, 1.0, 1.25, and $1.5\,\mathrm{m/s}$ in Figure 5 across the four flow directions, resulting in 24 conditions. We include speeds beyond $0.51\mathrm{m/s}$ to push the generalization boundary of the policy. In our evaluation, all conditions at $0.75\mathrm{m/s}$ and above result in zero SR. For $\mathbf{T} \times \mathbf{E2}$, we further evaluate densities of 500, 750, 1000, 1250, 1500, and $2000\,\mathrm{kg/m^3}$, resulting in 6 conditions in Figure 5. The densities of 500 and $750\,\mathrm{kg/m^3}$ cause the character to sink, while the density of $2000\,\mathrm{kg/m^3}$ results in low SR for asymmetric strokes such as front crawl and backstroke. We therefore exclude these boundary cases from the $\mathbf{E1}$ and $\mathbf{E2}$.

In the main paper, we use $\mathbf{T} \times (\mathbf{E1} + \mathbf{E2})$ for comprehensive generalization evaluation. For the ablation study and baseline comparison, we focus on the $\mathbf{T} \times \mathbf{E1}$ experiments, as they are sufficient to test policy robustness for comparison purposes.

A.3 Perturbation Generalization Settings

We evaluate robustness to two types of external perturbations: short-term impulses and continuous random forces. Impulses introduce brief, strong disturbances, while continuous random forces apply sustained disturbances. All perturbation forces are applied to the character root, allowing the disturbance to propagate through the whole body via articulated-body dynamics.

For impulses, we vary the force magnitude $\lambda \in 0.1, 0.25, 0.5, 1.0, 2.5, 5.0, 10.0 \times mg$, where m is character mass, six world directions ($\pm X$, $\pm Y$, $\pm Z$), and two application times: the mid-phase and end-phase of the reference cycle, giving 84 conditions. Each impulse is applied for 0.1s. For continuous forces, we sample forces from a Gaussian with zero mean and σ standard deviation at 100 Hz with $\sigma/mg \in [0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1.0]$, giving 7 conditions. In total, we use 91 perturbation conditions across all 30 base tasks, resulting in 2,730 episodes per policy per stroke.

A.4 Body Generalization Settings

We evaluate zero-shot adaptation to morphological and geometric changes across all 30 base tasks. For limb removal, we consider 8 variants: 4 bilateral symmetric removals (both arms, both forearms, both legs, and both shanks) and 4 unilateral removals (one arm, forearm, leg, or shank). For geometric changes, we add cylindrical foot fins of three lengths (10, 15, and 20 cm). In total, we use 11 conditions.

A.5 Metric Definitions and Justifications

Given the task above, we design metrics to evaluate how well the character follows the ideal trajectory. At the same time, we want the character's swimming motion to be natural. Therefore, we use both task metrics and motion quality metrics.

As mentioned in Section 4.1, we define eight metrics: five related to task performance and three related to motion quality. We first define **SRV** to evaluate whether the policy can prevent the character from flipping over or leaving the swimming pool, as survival is the most basic requirement. We then define **PS** to measure how closely the character follows the desired path. However, **PS** alone cannot distinguish between large deviation and lack of progress, as both can result in a large **PS**. We therefore use **Prog** to measure progress along the desired path over space and time, and **LD** to measure lateral deviation from the desired path. When **PS** is sufficiently small, these small differences are less important. We therefore also report **SR**, which measures whether the policy successfully follows the target trajectory. A motion is considered successful if the mean absolute error between the swimmer's root position and the intermediate goals is below a threshold of 0.2 m.

For motion quality, we aim for the generated motion to be natural. We therefore define three metrics. **Roll** measures whether the character's body roll remains within the range observed in human swimming. **Vel** measures how closely the joint velocities match those of the reference human swimming motion. However, a jittery motion can still have similar joint velocities to the reference motion. We therefore also use **Accel** to measure the smoothness of joint motion and detect excessive high-frequency movements.

For **Roll**, elite swimmers in front crawl and backstroke typically exhibit body roll of around 30–45° (Psycharakis & Sanders, 2010); we use 30° as the threshold. Butterfly and breaststroke are more symmetric, with much smaller body roll. As we did not find a reference specifying typical roll angles for butterfly and breaststroke, we use a small threshold of 10° for these two strokes.

We provide the detailed equations for each metric below. All metrics are computed per episode from frames sampled at 100Hz. We also skip the first 0.20s and compute each metric over the remaining evaluation window.

$$\mathcal{T} = \{\, t \mid 0.20\,\mathrm{s} \le t \le t_{\mathrm{end}} \,\}, \tag{10}$$

where t_{end} is the episode duration T_{ep}. SRV and SR are reported as percentages of episodes. The other metrics are computed per episode and then averaged across episodes.

Survival and success. For each experimental condition in Table 5, we run multiple simulated episodes. An evaluation episode is one rollout of the policy from a fixed initial state and task condition until the episode ends or is terminated early. An episode survives if it is not terminated early and the character never rolls over:

$$\mathrm{SRV}_i = \mathbb{I}\Big(\text{no early termination} \ \wedge \ \max_{t\in\mathcal{T}} |\theta_t^{\mathrm{roll}}| \le 90^\circ\Big), \tag{11}$$

where an episode is terminated early when the root leaves the pool through a wall or the floor, and θ_t^{roll} is the root roll angle defined below. An episode is considered successful if it survives and stays close to the desired trajectory:

$$\mathrm{SR}_i = \mathbb{I}\big(\mathrm{SRV}_i = 1 \ \wedge \ \mathrm{PS}_i \le \epsilon\big), \qquad \epsilon = 0.20\,\mathrm{m}. \tag{12}$$

SRV and **SR** are the percentages of surviving and successful episodes among the N episodes of a condition:

$$\mathrm{SRV} = \frac{100\%}{N}\sum_{i=1}^{N}\mathrm{SRV}_i, \qquad \mathrm{SR} = \frac{100\%}{N}\sum_{i=1}^{N}\mathrm{SR}_i. \tag{13}$$

Task metrics. **PS** is the mean distance between the root and the desired position:

$$\mathrm{PS} = \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\big\|\mathbf{p}_t^{\mathrm{root}} - \mathbf{p}_t^{\mathrm{traj}}\big\|_2. \tag{14}$$

$\mathbf{p}_t^{\text{root}}$ is the x-y plane position of the character root at frame t, and $\mathbf{p}_t^{\text{traj}}$ is the desired position at time t, which moves along the desired trajectory at constant speed and reaches its end at T_{ep}.

PS combines where the character is and when it gets there. **Prog** and **LD** separate these two aspects. Both use a progress coordinate $s(\mathbf{p})$ and a lateral distance $\ell(\mathbf{p})$ for a point $\mathbf{p}$. For straight lines, these are measured along and perpendicular to the desired trajectory from its start $\mathbf{p}^{\text{start}}$ to its end $\mathbf{p}^{\text{end}}$. With the unit direction $\hat{\mathbf{d}} = (\mathbf{p}^{\text{end}} - \mathbf{p}^{\text{start}})/|\mathbf{p}^{\text{end}} - \mathbf{p}^{\text{start}}|_2$,

$$s(\mathbf{p}) = (\mathbf{p} - \mathbf{p}^{\text{start}}) \cdot \hat{\mathbf{d}}, \qquad \ell(\mathbf{p}) = \big|(\mathbf{p} - \mathbf{p}^{\text{start}}) - s(\mathbf{p}), \hat{\mathbf{d}}\big|2. \tag{15}$$

For V-shaped polylines and sinusoidal trajectories, which do not have a single direction, $s(\mathbf{p})$ is the arc length from the start of the trajectory to the point on it closest to $\mathbf{p}$, and $\ell(\mathbf{p})$ is the distance to that point. Measuring a V against its own two segments rather than against the straight line from start to end matters: the corner of a V lies off that line, so a character following the V exactly would otherwise be charged a lateral deviation it did not commit, up to $0.2\,\text{m}$ for the sharpest V over a $3\,\text{m}$ swim. We then compute:

$$\text{Prog} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{s(\mathbf{p}_t^{\text{root}})}{s(\mathbf{p}_t^{\text{traj}})}, \qquad \text{LD} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \ell(\mathbf{p}_t^{\text{root}}). \tag{16}$$

A **Prog** value close to 1 means that the character keeps pace with the desired trajectory; a value below 1 means it falls behind, while a value above 1 means it moves ahead. Skipping the first 0.20s also keeps the denominator of **Prog** away from zero.

Motion quality. **Roll** is the mean root roll beyond a style-specific tolerance:

$$\text{Roll} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \max\big(0,\ |\theta_t^{\text{roll}}| - \theta_{\text{tol}}\big), \tag{17}$$

where θ_t^{roll} is the root roll angle of the character and $\theta_{\text{tol}} = 30^\circ$ for front crawl and backstroke and 10° for breaststroke and butterfly.

Vel is the mean difference between the joint velocities of the character and the reference motion:

$$\text{Vel} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{1}{D_q} \sum_{j=1}^{D_q} \big|\dot{q}_{t,j} - \dot{q}_{t,j}^{\text{ref}}\big|, \qquad \dot{\mathbf{q}}_t^{\text{ref}} = \omega_t\, \dot{\mathbf{q}}^{\text{ref}}(\phi_t), \tag{18}$$

where D_q is the number of actuated joint DoFs (36 for the full character). The reference velocity is sampled at the policy's phase ϕ_t and scaled by its stroke frequency ω_t (Section 3.4).

Accel is the mean joint acceleration and does not use a reference motion:

$$\text{Accel} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{1}{D_q} \sum_{j=1}^{D_q} \big|\ddot{q}_{t,j}\big|, \qquad \ddot{q}_{t,j} = \frac{\dot{q}_{t,j} - \dot{q}_{t-1,j}}{\Delta t}, \tag{19}$$

where $\Delta t = 0.01\,\text{s}$. Lower values indicate smoother motion, while higher values indicate more jittery control.

Recovery from disturbances. We additional introduce metrics for impulse perturbation that is to evaluate how large LD can be and how fast character can swim back to the desired path. An impulse perturbation starts at t_{imp} and lasts Δt_{imp}, we report the largest lateral deviation LD_{peak} after the impulse, and the time the character needs to return to the desired path tolerance $\ell \le \epsilon_{\text{rec}} = 0.08\,\text{m}$, where ℓ as mentioned in Equation (15):

$$\text{LD}_{\text{peak}} = \max_{t \ge t_{\text{imp}}} \ell(\mathbf{p}_t^{\text{root}}), \qquad \text{T}_{\text{recover}} = t_{\text{settle}} - t_{\text{imp}}, \tag{20}$$

$$t_{\text{settle}} = \min\Big\{\, t \ge t_{\text{imp}} + \Delta t_{\text{imp}} \;\Big|\; \ell(\mathbf{p}_{t'}^{\text{root}}) \le \epsilon_{\text{rec}} \ \ \forall\, t' \ge t \,\Big\}. \tag{21}$$

If the character does not return to the path within the tolerance, we count the experiment as not recovered and report no T_{recover}. We still report LD_{peak} for these episodes as the largest deviation measured before termination. Episodes that terminate before the impulse starts are excluded. We therefore report three quantities: LD_{peak} averaged over all perturbed episodes, T_{recover} averaged over recovered episodes only, and the percentage of recovered episodes, in Figure 21. For continuous perturbation, we report only LD_{peak}.

A.6 Reference Motion Data

We will release the Reference Motion Data upon acceptance.

Unlike previous human motion work (Peng et al., 2018; Xu et al., 2025; Shi et al., 2026), there's barely motion capture data for swimming, therefore we construct a dataset of four swimming strokes, represented as local joint angle sequences $\mathbf{q}^{\text{ref}} \in \mathbb{R}^{T\times 36}$ for 12 joints. The motion is sampled at 100 Hz, and each stroke is represented as a looped sequence. The cycle durations are 1.82 s for front crawl, 1.75 s for backstroke, 1.52 s for breaststroke, and 1.36 s for butterfly, resulting in 644 frames (6.44 s) in total. The motion data are preprocessed from swimming motion-capture recordings (LABIOMEP & Qualisys, 2025). In this section, we describe our preprocessing procedure.

We preprocess the raw motion capture data in two stages: spatial reconstruction and cleanup, followed by temporal periodization and smoothing. The goal is to obtain a kinematically consistent and periodic reference motion that can be used for physics-based control.

Spatial reconstruction and cleanup The four strokes exhibit different body sizes, although all motions were captured from the same subject. We therefore construct a shared skeletal rig by averaging the bone lengths across the four strokes and use it as the human model. For each joint, the local rotation is then computed from the global rotations of the parent and child joints as $R_{\text{local}} = R_{\text{parent}}^{-1} R_{\text{child}}$. where R is the rotation. Finally, to remove isolated high-frequency artifacts, we smooth the recovered rotations on the $SO(3)$ manifold using a short-window Gaussian filter.

Temporal Periodization and Regularization. After spatial cleanup, we extract one stroke cycle from each captured sequence. We select the cycle boundaries by minimizing the mismatch in joint orientations and velocities, using geodesic distances for rotations. For joints with large ranges of motion, we re-center the joint-angle representation around the mean pose of the selected cycle to avoid coordinate singularities. We then apply a short boundary ramp over the final 10 frames to remove the remaining discontinuity at the cycle boundary.

A.7 Training vs Generalization setup

Training setup As mentioned in Section 3.6, to reduce the training time required for physical simulation, we use a small swimming pool for training. We set the episode duration to 4.0 s, which contains at least two cycles of the reference motion. We set the swimming distance to 1.5 m, which corresponds over 4.0 s of motion for our 1.2 m-tall character (Pendergast et al., 1977) .

As shown in Figure 11, all policies are trained in the small pool. The character starts from a fixed initial state at the fixed position $(0, 0, 0)$ and swims toward one of three target goals:

$$\mathbf{p}^{\text{tar}} = (1.5\text{ m},\ d_{\text{lat}},\ 0), \qquad d_{\text{lat}} \in \{0, \pm 0.375\text{ m}\}.$$

These correspond to a straight target and two angled targets with approximately $\theta_{\text{lat}} = 0^\circ, \pm 14^\circ$.

Generalization setup As shown in Figure 11, for the larger evaluation pools, we use pools with $2\times$ or $4\times$ the forward swimming distance of the training pool. The larger pools require more SPH particles and therefore more computation.

At evaluation time, following Peng et al. (2018); Zhang et al. (2025); Peng (2025), we set the policy's action standard deviation to zero and use the resulting deterministic action. For experiments metrics, we report the mean and standard deviation over 3 runs. For all ablation and comparison experiments, we train each policy with three different seeds, evaluate each trained policy at 10M samples, and report the mean and standard deviation across seeds.

For goal-state update during training and inference, unlike previous work, we do not have a global reference trajectory for swimming. Instead, we define a target swimming speed as the swimming distance divided by the required motion time. During training, we sample lookahead goals along the straight line from the initial position to the final target based on this target speed. During inference, for non-linear trajectory following, we update the goal state along the local tangent direction of the planned path while maintaining the same target swimming speed, as shown in Figure 12.

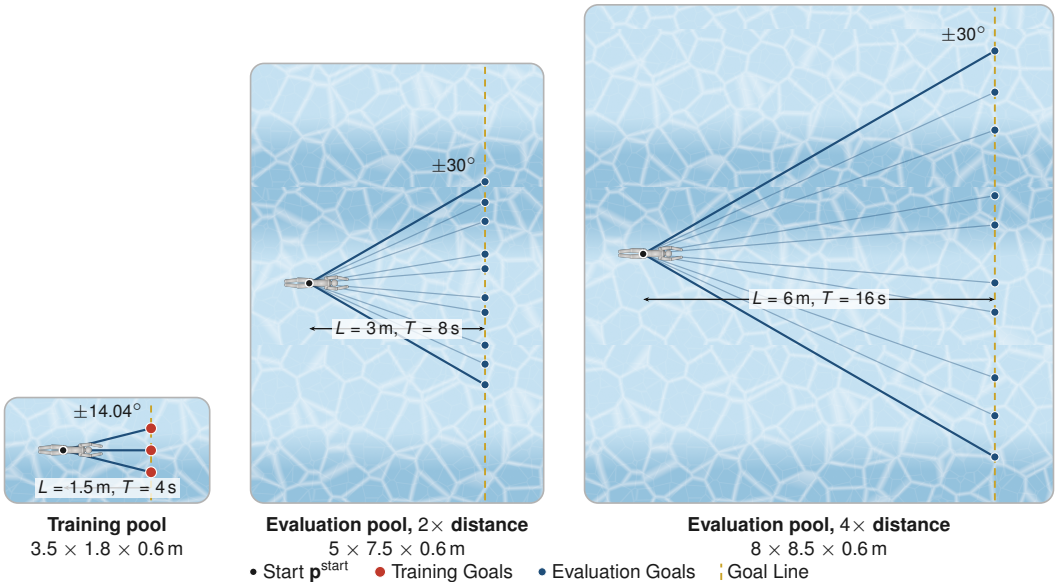

Figure 11: The training and evaluation pools. Top view, with the character shown at true scale. Policies are trained in the small pool and evaluated in two larger pools with $2\times$ and $4\times$ the forward swimming distance. Both evaluation pools are shown in goal-reaching task. The character starts 1.0 m from the near wall, and targets lie 1.0 m from the far wall.

B Additional Experimental Results

B.1 Detailed and Additional Generalization

B.1.1 Qualitative Results

We show visual results for four swimming styles (Figure 8), under challenging conditions (Figure 9). Surprisingly, we find that a 0.5 mg impulse perturbation improve trajectory following of 'R'. Our intuition is that this perturbation place the character behind the target, encouraging the policy to follow the goal more closely.

We also show the top views of goal-reaching and trajectory-following in Figure 13, together with an example of complex control trajectories in Figure 7, and compare SWIM with the baselines on the same task in Figure 14. Videos are in the supplementary material. Overall, they show SWIM, despite trained on simple task and settings, can generalize to drastically different and challenging unseen conditions.

B.1.2 Training Reward Details

Figure 15 shows the training curves for our policy across three seeds. Overall, policy learning is stable. When the forward and deviation rewards exceed 0.8, the root and intermediate target tracking errors are below 0.1 m. Thus, the policy has already learned to solve the training task well by 5M samples. We therefore use 10M samples for all training settings to provide a fair comparison.

During the early stage of training, within the first 1M samples, the policy primarily learns to maintain stability and increase swimming speed. In the first half of training, it further learns to reduce trajectory deviation while continuing to improve swimming speed.

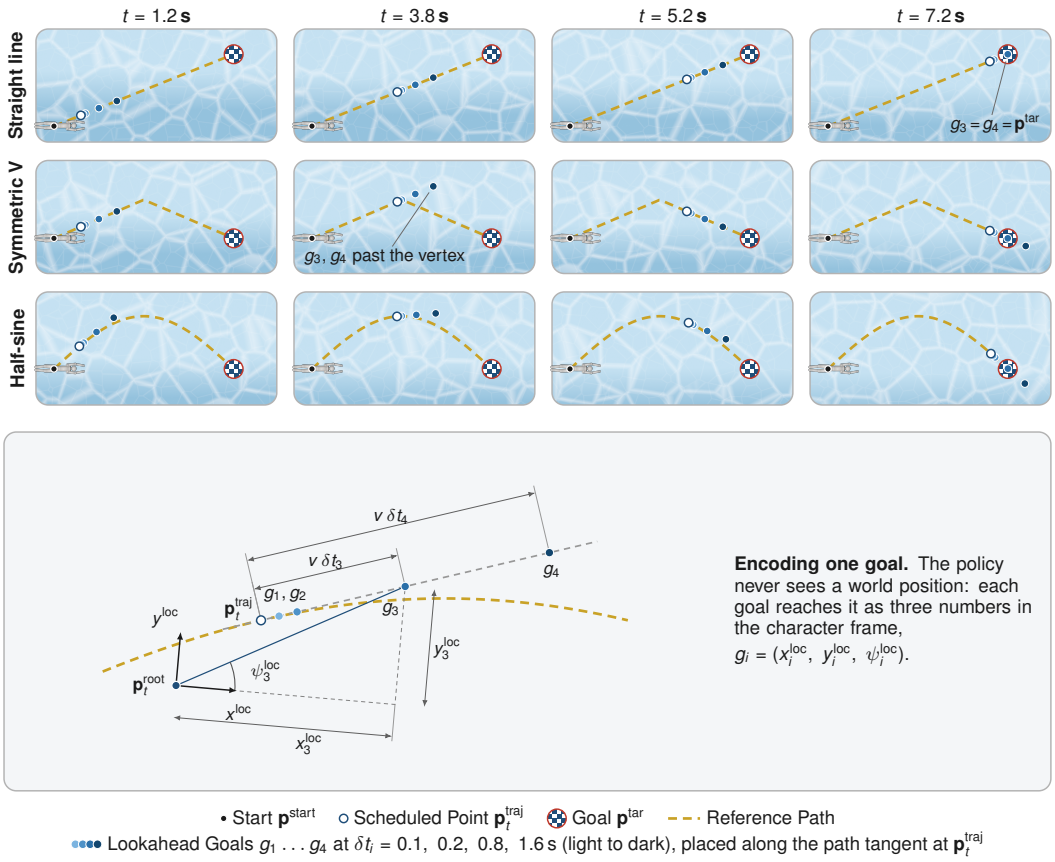

Figure 12: **Goal states over an episode for three trajectories.** Rows show straight-line, symmetric V-shaped, and half-sine trajectories at the 2× evaluation setting ($L = 3$ m, $T_{\text{ep}} = 8$ s); columns show four time points. Goal states are placed δt_i seconds ahead along the local path tangent, with distance $v, \delta t_i$. Each goal is represented in the character local system by $(x_i^{\text{loc}}, y_i^{\text{loc}}, \psi_i^{\text{loc}})$, rather than its world position. **Bottom:** Zoomed view of the half-sine at $t = 3$ s with equal axes. The character pose is illustrative; its actual position is determined by the policy.

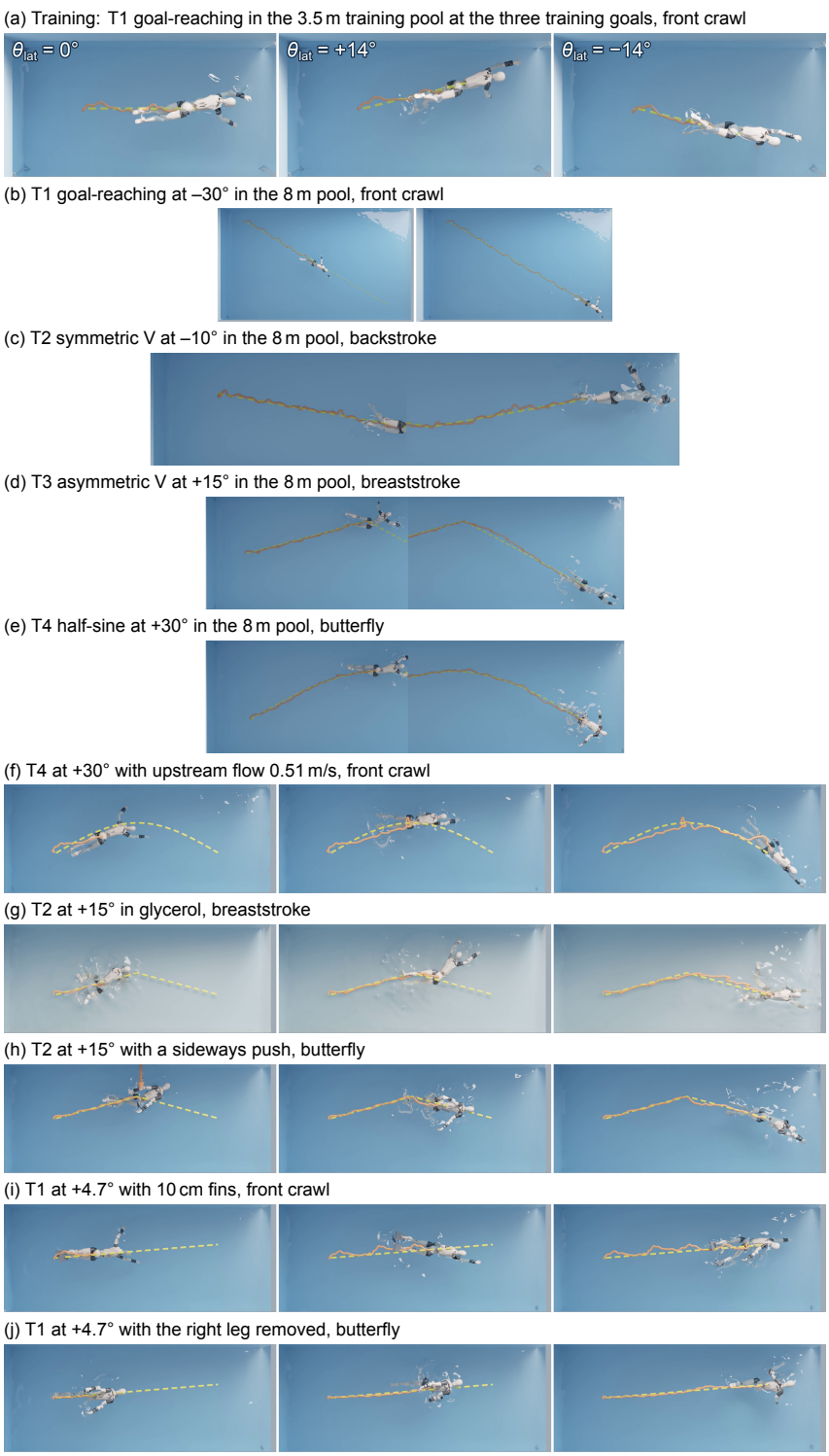

Figure 13: **Tasks and conditions, top view.** (a): the training task, goal-reaching in still water in the small training pool at the three training goals, one episode each at its end (4 s). The other rows are one SWIM episode each. (b) to (e): the four tasks in still water, one per stroke, in the unseen 8 m pool; each shows the whole pool with the swimmer at 8 and 16 s. (f) to (j): one unseen condition each, at about 2, 5 and 8 s, except (h), which is during the push at about 4.2 s, about 1 s later and at 8 s. Dashed yellow: target path. Orange: swum path. Every episode shown reaches its goal.

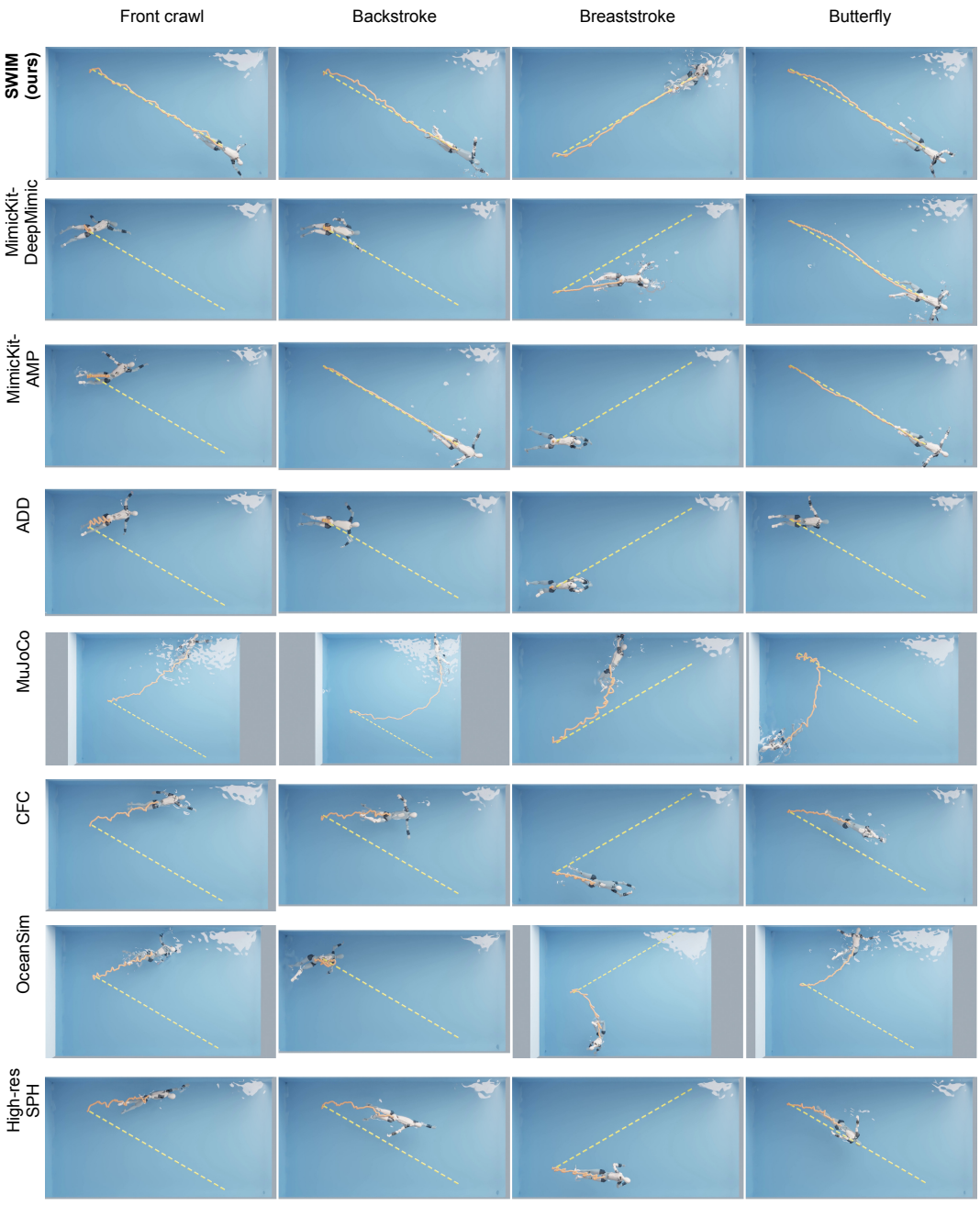

Figure 14: **SWIM and the baselines on the same task.** T1 goal-reaching at 30° in still water (−30°, and +30° for breaststroke), about twice the largest training angle, shown at the end of the episode (8 s). All policies are run in our SPH fluid. Dashed yellow: the straight path to the goal. Orange: the path swum. SWIM reaches the goal in all four strokes, MimicKit-AMP in backstroke and butterfly, and MimicKit-DeepMimic in butterfly; no other method reaches it. Some MuJoCo and OceanSim swimmers leave the lane, so their pool is drawn wider.

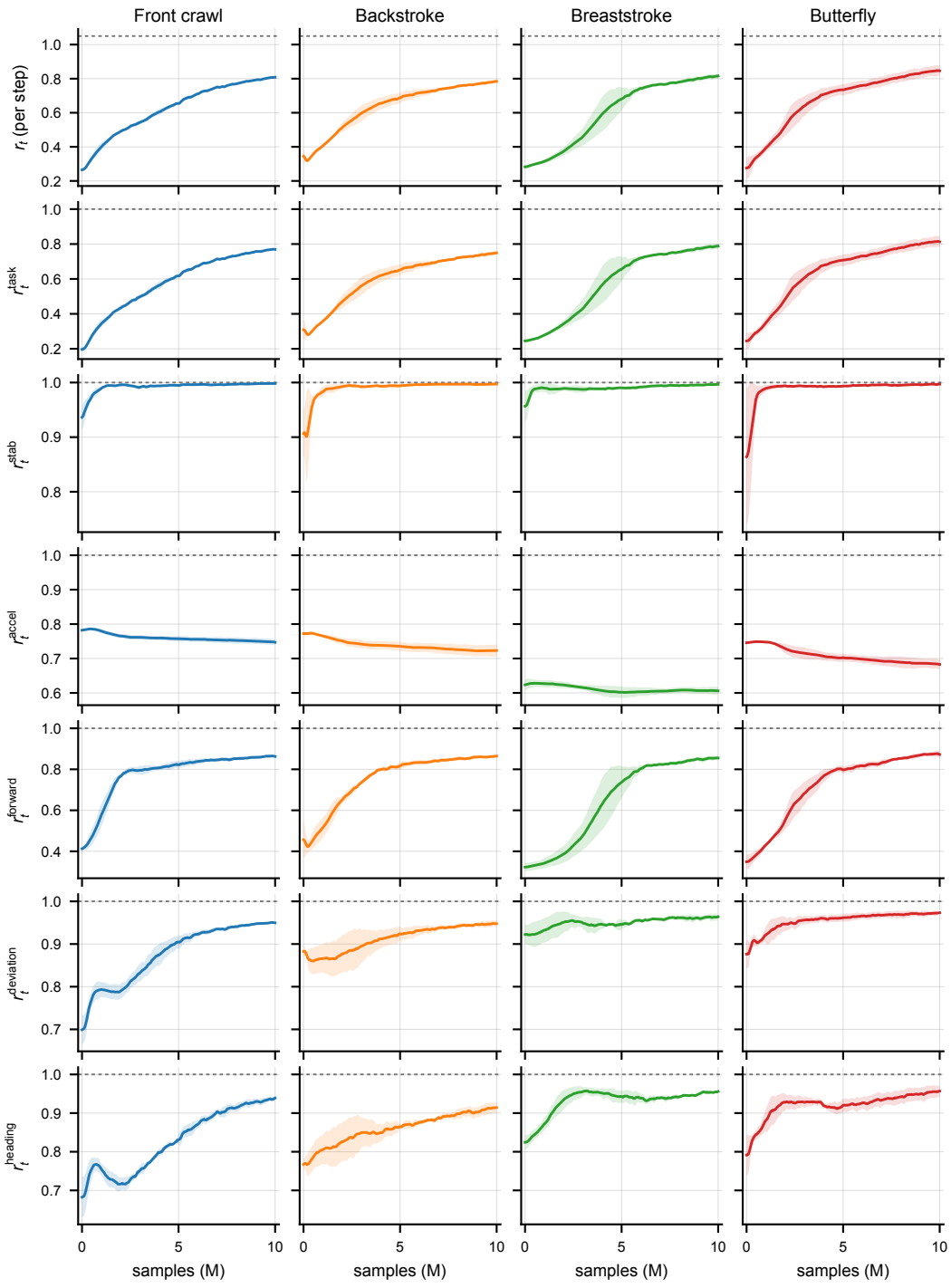

Figure 15: **Training reward of SWIM.** Rows show the individual reward terms, and columns show the four swimming strokes. Lines represent the mean over three seeds, and bands represent one standard deviation. The dotted line indicates the maximum possible value of each quantity. Rewards are normalized.

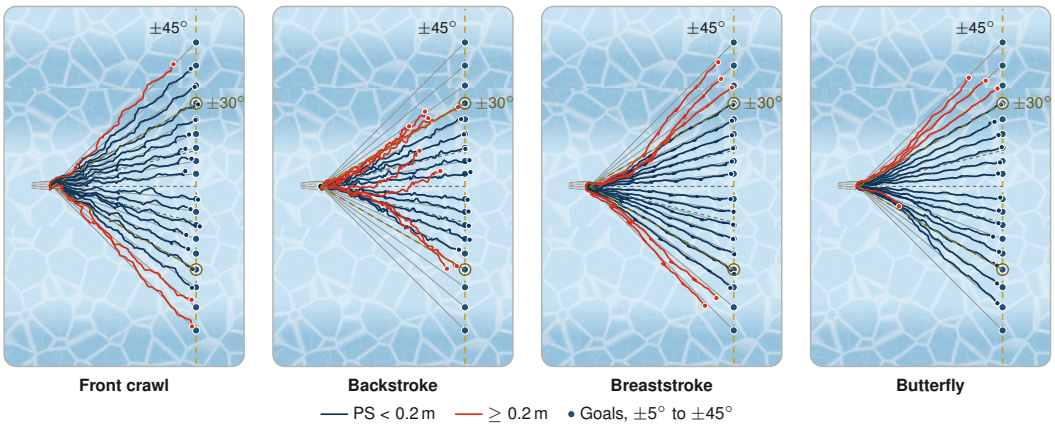

Figure 16: Trajectories corresponding to Figure 5. A trajectory is shown in blue when it ends within PS $\leq$ 0.20m and in red otherwise.

Table 7: **Goal-reaching in the 8 m pool**

Bearing	Stroke	SRV↑ (%)	SR↑ (%)	PS↓ (m)	LD↓ (m)	Prog→ 1	Roll↓ (°)	Vel↓ (rad/s)	Accel↓ (rad/s^2)
±4.7°	Front crawl	100.0	100.0	$0.082_{\pm 0.011}$	$0.032_{\pm 0.000}$	$0.965_{\pm 0.001}$	$0.49_{\pm 0.11}$	$0.97_{\pm 0.03}$	$52.16_{\pm 1.06}$
	Backstroke	100.0	100.0	$0.068_{\pm 0.006}$	$0.021_{\pm 0.002}$	$0.974_{\pm 0.002}$	$0.29_{\pm 0.02}$	$1.19_{\pm 0.01}$	$54.79_{\pm 0.50}$
	Breaststroke	100.0	100.0	$0.058_{\pm 0.002}$	$0.027_{\pm 0.001}$	$0.983_{\pm 0.006}$	$0.08_{\pm 0.00}$	$1.53_{\pm 0.00}$	$87.82_{\pm 1.40}$
	Butterfly	100.0	100.0	$0.035_{\pm 0.005}$	$0.013_{\pm 0.003}$	$0.980_{\pm 0.002}$	$0.04_{\pm 0.05}$	$1.51_{\pm 0.01}$	$73.99_{\pm 0.15}$
±9.4°	Front crawl	100.0	100.0	$0.093_{\pm 0.004}$	$0.027_{\pm 0.000}$	$0.959_{\pm 0.002}$	$0.51_{\pm 0.07}$	$0.98_{\pm 0.00}$	$52.58_{\pm 0.01}$
	Backstroke	100.0	100.0	$0.075_{\pm 0.008}$	$0.025_{\pm 0.002}$	$0.985_{\pm 0.007}$	$0.12_{\pm 0.05}$	$1.18_{\pm 0.00}$	$54.82_{\pm 0.01}$
	Breaststroke	100.0	100.0	$0.061_{\pm 0.003}$	$0.024_{\pm 0.001}$	$0.973_{\pm 0.005}$	$0.10_{\pm 0.04}$	$1.54_{\pm 0.02}$	$88.64_{\pm 1.56}$
	Butterfly	100.0	100.0	$0.035_{\pm 0.005}$	$0.013_{\pm 0.000}$	$0.977_{\pm 0.002}$	$0.03_{\pm 0.00}$	$1.53_{\pm 0.01}$	$74.61_{\pm 0.42}$
±19.4°	Front crawl	100.0	100.0	$0.094_{\pm 0.015}$	$0.029_{\pm 0.002}$	$0.944_{\pm 0.003}$	$0.89_{\pm 0.01}$	$1.01_{\pm 0.01}$	$53.21_{\pm 1.56}$
	Backstroke	100.0	100.0	$0.076_{\pm 0.006}$	$0.023_{\pm 0.006}$	$0.949_{\pm 0.005}$	$0.23_{\pm 0.29}$	$1.22_{\pm 0.00}$	$55.44_{\pm 0.44}$
	Breaststroke	100.0	100.0	$0.064_{\pm 0.001}$	$0.028_{\pm 0.003}$	$0.961_{\pm 0.016}$	$0.08_{\pm 0.07}$	$1.56_{\pm 0.00}$	$87.54_{\pm 1.31}$
	Butterfly	100.0	100.0	$0.049_{\pm 0.003}$	$0.017_{\pm 0.000}$	$0.956_{\pm 0.015}$	$0.14_{\pm 0.01}$	$1.58_{\pm 0.01}$	$76.69_{\pm 0.95}$
±24.7°	Front crawl	100.0	100.0	$0.084_{\pm 0.007}$	$0.033_{\pm 0.001}$	$0.942_{\pm 0.007}$	$0.10_{\pm 0.06}$	$1.01_{\pm 0.03}$	$52.25_{\pm 1.55}$
	Backstroke	100.0	100.0	$0.114_{\pm 0.052}$	$0.029_{\pm 0.001}$	$0.929_{\pm 0.045}$	$0.41_{\pm 0.21}$	$1.23_{\pm 0.01}$	$55.05_{\pm 0.15}$
	Breaststroke	100.0	100.0	$0.084_{\pm 0.022}$	$0.029_{\pm 0.001}$	$0.946_{\pm 0.025}$	$0.27_{\pm 0.10}$	$1.59_{\pm 0.01}$	$89.07_{\pm 1.59}$
	Butterfly	100.0	100.0	$0.053_{\pm 0.007}$	$0.017_{\pm 0.004}$	$0.957_{\pm 0.017}$	$0.32_{\pm 0.28}$	$1.58_{\pm 0.01}$	$77.63_{\pm 0.63}$
±30.0°	Front crawl	100.0	100.0	$0.087_{\pm 0.005}$	$0.033_{\pm 0.001}$	$0.935_{\pm 0.003}$	$0.15_{\pm 0.01}$	$1.06_{\pm 0.02}$	$53.58_{\pm 1.40}$
	Backstroke	100.0	100.0	$0.161_{\pm 0.033}$	$0.048_{\pm 0.032}$	$0.899_{\pm 0.033}$	$0.22_{\pm 0.18}$	$1.26_{\pm 0.01}$	$56.58_{\pm 0.08}$
	Breaststroke	100.0	100.0	$0.093_{\pm 0.015}$	$0.036_{\pm 0.011}$	$0.933_{\pm 0.007}$	$0.07_{\pm 0.03}$	$1.60_{\pm 0.00}$	$88.95_{\pm 0.94}$
	Butterfly	100.0	100.0	$0.099_{\pm 0.011}$	$0.024_{\pm 0.005}$	$0.938_{\pm 0.036}$	$0.14_{\pm 0.03}$	$1.64_{\pm 0.01}$	$80.03_{\pm 1.09}$

B.1.3 Goal-reaching Details

Boundary analysis Figure 16 shows the trajectories corresponding to Figure 5 a. For front crawl, breaststroke, and butterfly, some trajectories follow the ideal paths but are not counted as successful cases. This demonstrates that our success threshold is strict, as mentioned in Section A.5.

Four stroke & Time slice We further test the 10 unseen goal-reaching tasks in a larger, unseen swimming pool with $4\times$ the forward distance, as described in Section A.7. The results are shown in Table 7. Overall, in the larger swimming pool, SWIM achieves 100% SRV and SR across all 10 tasks, outperforming the corresponding $2\times$ setting shown in Figure 16. Although the turning angles remain the same, the longer motion duration in the larger swimming pool provides more time for the policy to adjust its trajectory, since the swimming motion can primarily adjust the forward direction after each stroke phase as discussed in Section 4.2.

We also analyze the temporal differences in the swimming motion. During the first second, Figure 17 shows the policy achieves only 0.4-0.6 progress while exhibiting higher velocity and acceleration. This is because the character starts from zero velocity and needs to accelerate to swimming speed. This shows that the policy has learned how to accelerate from rest to its normal swimming speed.

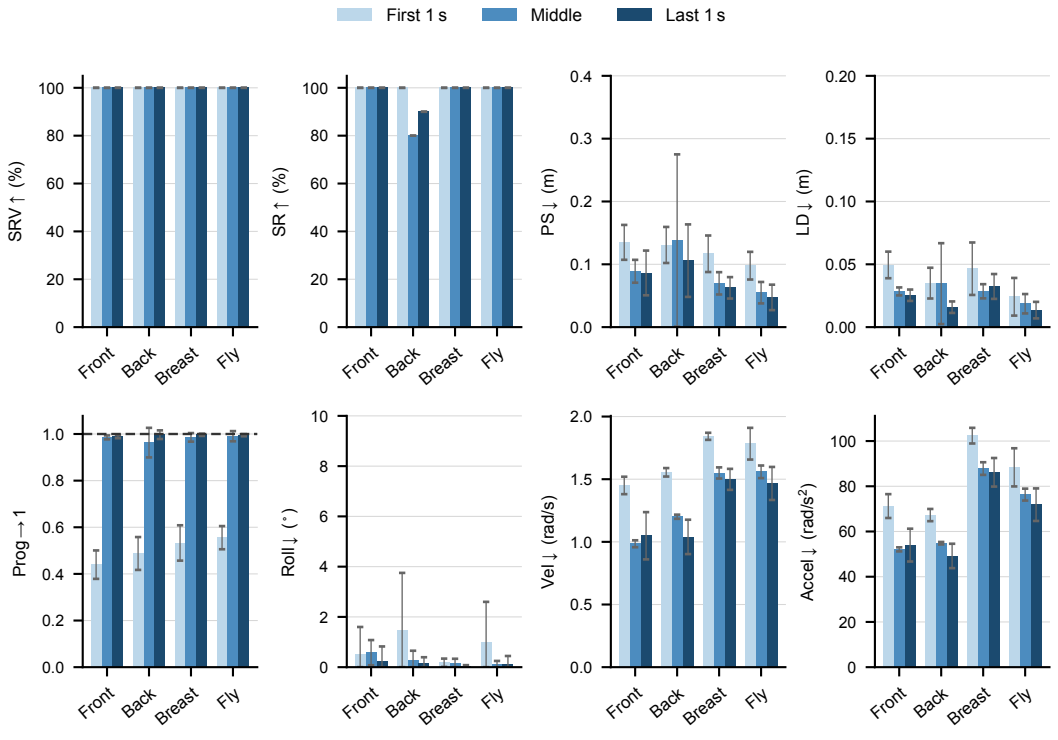

Figure 17: The eight metrics over the first, middle, and last seconds of the goal-reaching tasks in the 8m pool (Table 7)

Table 8: **Interpolated vs extrapolated goals.** The ten goals of the 8 m pool (Table 7)

Range	Stroke	SRV↑ (%)	SR↑ (%)	PS↓ (m)	LD↓ (m)	Prog→ 1	Roll↓ (°)	Vel↓ (rad/s)	Accel↓ (rad/s^2)
Interpolated ($\lvert\theta_{lat}\rvert \leq 14.04°$)	Front crawl	100.0	100.0	$0.087_{\pm 0.009}$	$0.029_{\pm 0.003}$	$0.962_{\pm 0.004}$	$0.50_{\pm 0.08}$	$0.98_{\pm 0.02}$	$52.37_{\pm 0.66}$
	Backstroke	100.0	100.0	$0.071_{\pm 0.007}$	$0.023_{\pm 0.003}$	$0.980_{\pm 0.007}$	$0.21_{\pm 0.10}$	$1.18_{\pm 0.01}$	$54.80_{\pm 0.29}$
	Breaststroke	100.0	100.0	$0.060_{\pm 0.002}$	$0.026_{\pm 0.002}$	$0.978_{\pm 0.007}$	$0.09_{\pm 0.03}$	$1.54_{\pm 0.01}$	$88.23_{\pm 1.30}$
	Butterfly	100.0	100.0	$0.035_{\pm 0.004}$	$0.013_{\pm 0.002}$	$0.978_{\pm 0.002}$	$0.04_{\pm 0.03}$	$1.52_{\pm 0.01}$	$74.30_{\pm 0.44}$
Extrapolated ($\lvert\theta_{lat}\rvert > 14.04°$)	Front crawl	100.0	100.0	$0.088_{\pm 0.009}$	$0.031_{\pm 0.002}$	$0.940_{\pm 0.006}$	$0.38_{\pm 0.40}$	$1.03_{\pm 0.03}$	$53.01_{\pm 1.32}$
	Backstroke	100.0	100.0	$0.117_{\pm 0.047}$	$0.033_{\pm 0.019}$	$0.926_{\pm 0.034}$	$0.29_{\pm 0.20}$	$1.23_{\pm 0.02}$	$55.69_{\pm 0.74}$
	Breaststroke	100.0	100.0	$0.080_{\pm 0.018}$	$0.031_{\pm 0.007}$	$0.947_{\pm 0.019}$	$0.14_{\pm 0.11}$	$1.58_{\pm 0.02}$	$88.52_{\pm 1.27}$
	Butterfly	100.0	100.0	$0.067_{\pm 0.026}$	$0.019_{\pm 0.004}$	$0.950_{\pm 0.021}$	$0.20_{\pm 0.16}$	$1.60_{\pm 0.03}$	$78.12_{\pm 1.69}$

Interpolation Vs extrapolation Table 8 shows that the average PS errors across the four strokes are 0.087m, 0.071m, 0.060m, and 0.035m for interpolation, and 0.088m, 0.117m, 0.080m, and 0.067m for extrapolation, respectively. This shows extrapolation has slightly larger PS errors and indicates that extrapolation is more challenging for the policy than interpolation.

Table 9: **Trajectory-following in the 8 m pool**

Shape	Stroke	SRV↑ (%)	SR↑ (%)	PS↓ (m)	LD↓ (m)	Prog→ 1	Roll↓ (°)	Vel↓ (rad/s)	Accel↓ (rad/s^2)
Symmetric V-shapes	Front crawl	100.0	100.0	$0.103_{\pm 0.029}$	$0.038_{\pm 0.011}$	$0.957_{\pm 0.010}$	$0.69_{\pm 0.23}$	$0.99_{\pm 0.02}$	$52.94_{\pm 0.70}$
	Backstroke	100.0	75.0	$0.188_{\pm 0.270}$	$0.099_{\pm 0.187}$	$0.955_{\pm 0.046}$	$0.44_{\pm 0.67}$	$1.18_{\pm 0.03}$	$53.62_{\pm 1.67}$
	Breaststroke	100.0	100.0	$0.079_{\pm 0.024}$	$0.045_{\pm 0.027}$	$0.981_{\pm 0.011}$	$0.37_{\pm 0.27}$	$1.55_{\pm 0.03}$	$88.34_{\pm 2.12}$
	Butterfly	87.5	87.5	$0.084_{\pm 0.086}$	$0.040_{\pm 0.057}$	$0.970_{\pm 0.014}$	$0.94_{\pm 2.45}$	$1.54_{\pm 0.03}$	$75.45_{\pm 1.26}$
Asymmetric V-shapes	Front crawl	100.0	100.0	$0.088_{\pm 0.009}$	$0.032_{\pm 0.002}$	$0.957_{\pm 0.007}$	$0.34_{\pm 0.11}$	$1.00_{\pm 0.01}$	$52.62_{\pm 0.46}$
	Backstroke	100.0	50.0	$0.287_{\pm 0.312}$	$0.171_{\pm 0.242}$	$0.937_{\pm 0.051}$	$0.63_{\pm 0.61}$	$1.17_{\pm 0.03}$	$53.37_{\pm 1.64}$
	Breaststroke	100.0	100.0	$0.083_{\pm 0.041}$	$0.052_{\pm 0.044}$	$0.986_{\pm 0.007}$	$0.30_{\pm 0.22}$	$1.55_{\pm 0.01}$	$88.02_{\pm 0.55}$
	Butterfly	100.0	100.0	$0.048_{\pm 0.009}$	$0.017_{\pm 0.003}$	$0.973_{\pm 0.009}$	$0.15_{\pm 0.09}$	$1.54_{\pm 0.03}$	$75.55_{\pm 1.17}$
Sinusoidal trajectories	Front crawl	100.0	100.0	$0.091_{\pm 0.013}$	$0.034_{\pm 0.005}$	$0.952_{\pm 0.010}$	$0.44_{\pm 0.20}$	$1.01_{\pm 0.03}$	$53.20_{\pm 0.69}$
	Backstroke	100.0	100.0	$0.098_{\pm 0.043}$	$0.044_{\pm 0.024}$	$0.955_{\pm 0.031}$	$0.46_{\pm 0.46}$	$1.20_{\pm 0.02}$	$54.55_{\pm 0.51}$
	Breaststroke	100.0	100.0	$0.071_{\pm 0.012}$	$0.029_{\pm 0.007}$	$0.969_{\pm 0.022}$	$0.16_{\pm 0.09}$	$1.55_{\pm 0.02}$	$88.32_{\pm 0.79}$
	Butterfly	100.0	100.0	$0.049_{\pm 0.013}$	$0.016_{\pm 0.002}$	$0.970_{\pm 0.018}$	$0.14_{\pm 0.07}$	$1.55_{\pm 0.03}$	$75.68_{\pm 1.67}$

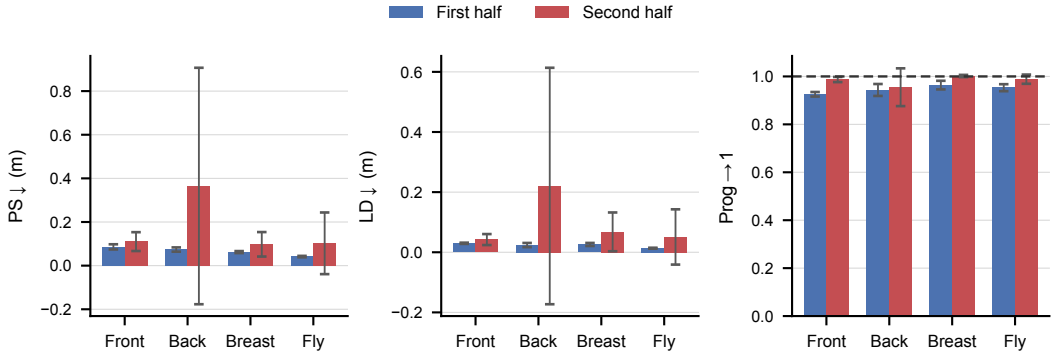

Figure 18: **Before and after the sharp turn of the trajectory-following task.** The 12 V-shaped trajectories in the 8 m pool.

B.1.4 Trajectory-following Details

We further evaluate trajectory-following performance in a pool with $4\times$ the swimming distance, as described in Section A.7. Table 9 shows the results for three trajectory shapes. Sharp turns are more challenging, particularly for backstroke: the PS error increases to 0.188 m for the symmetric V-shapes and 0.287 m for the asymmetric V-shapes, compared with 0.098 m for the sinusoidal trajectories. The corresponding SR also decreases from 100% for the sinusoidal trajectories to 75% and 50%, respectively. These results indicate that sharp turns pose greater challenges for trajectory-following, especially for backstroke.

Sharp V-turn findings We further analyze the reasons for the lower performance on sharp turns. Figure 18 shows that, in the second half of the task, lateral deviation errors are large, while progress remains around 0.9. This indicates that lateral deviation is the main cause of failure. Our intuition is that, although the maximum turning angle is around $30°$, the character may over-turn during the first turn and face away from the direction of the second segment, effectively requiring a larger turning angle.

We also report detailed metrics for the sinusoidal trajectory. Table 10 shows that the policy achieves good performance on this continuously changing trajectory. This further supports our intuition about sharp turns, as the maximum turning angle is also around $30°$ for the sinusoidal trajectory.

Table 10: **The sinusoidal trajectories by amplitude.**

Amplitude	Stroke	SRV↑ (%)	SR↑ (%)	PS↓ (m)	LD↓ (m)	Prog→ 1	Roll↓ (°)	Vel↓ (rad/s)	Accel↓ (rad/s^2)
$A = \pm 0.276$ m	Front crawl	100.0	100.0	$0.088_{\pm 0.006}$	$0.031_{\pm 0.004}$	$0.960_{\pm 0.003}$	$0.55_{\pm 0.18}$	$0.98_{\pm 0.00}$	$52.26_{\pm 0.02}$
	Backstroke	100.0	100.0	$0.069_{\pm 0.003}$	$0.029_{\pm 0.000}$	$0.975_{\pm 0.002}$	$0.19_{\pm 0.17}$	$1.20_{\pm 0.01}$	$54.96_{\pm 0.17}$
	Breaststroke	100.0	100.0	$0.068_{\pm 0.000}$	$0.024_{\pm 0.005}$	$0.988_{\pm 0.012}$	$0.14_{\pm 0.00}$	$1.53_{\pm 0.03}$	$87.68_{\pm 1.17}$
	Butterfly	100.0	100.0	$0.040_{\pm 0.001}$	$0.015_{\pm 0.003}$	$0.984_{\pm 0.005}$	$0.08_{\pm 0.08}$	$1.52_{\pm 0.01}$	$74.19_{\pm 0.23}$
$A = \pm 0.550$ m	Front crawl	100.0	100.0	$0.089_{\pm 0.017}$	$0.031_{\pm 0.001}$	$0.963_{\pm 0.005}$	$0.65_{\pm 0.15}$	$1.01_{\pm 0.02}$	$53.36_{\pm 0.75}$
	Backstroke	100.0	100.0	$0.069_{\pm 0.005}$	$0.024_{\pm 0.001}$	$0.973_{\pm 0.020}$	$0.07_{\pm 0.09}$	$1.19_{\pm 0.01}$	$54.63_{\pm 0.63}$
	Breaststroke	100.0	100.0	$0.062_{\pm 0.005}$	$0.028_{\pm 0.006}$	$0.975_{\pm 0.016}$	$0.07_{\pm 0.00}$	$1.54_{\pm 0.01}$	$88.20_{\pm 0.68}$
	Butterfly	100.0	100.0	$0.041_{\pm 0.001}$	$0.014_{\pm 0.003}$	$0.971_{\pm 0.007}$	$0.14_{\pm 0.04}$	$1.54_{\pm 0.02}$	$74.78_{\pm 1.26}$
$A = \pm 0.826$ m	Front crawl	100.0	100.0	$0.087_{\pm 0.016}$	$0.032_{\pm 0.001}$	$0.944_{\pm 0.006}$	$0.32_{\pm 0.01}$	$1.03_{\pm 0.01}$	$53.55_{\pm 0.16}$
	Backstroke	100.0	100.0	$0.089_{\pm 0.025}$	$0.045_{\pm 0.013}$	$0.958_{\pm 0.017}$	$0.59_{\pm 0.24}$	$1.21_{\pm 0.02}$	$54.24_{\pm 0.32}$
	Breaststroke	100.0	100.0	$0.070_{\pm 0.010}$	$0.028_{\pm 0.010}$	$0.969_{\pm 0.026}$	$0.15_{\pm 0.03}$	$1.56_{\pm 0.00}$	$88.20_{\pm 0.22}$
	Butterfly	100.0	100.0	$0.047_{\pm 0.002}$	$0.015_{\pm 0.003}$	$0.969_{\pm 0.015}$	$0.18_{\pm 0.14}$	$1.56_{\pm 0.02}$	$75.87_{\pm 1.08}$
$A = \pm 1.100$ m	Front crawl	100.0	100.0	$0.101_{\pm 0.018}$	$0.040_{\pm 0.006}$	$0.943_{\pm 0.004}$	$0.24_{\pm 0.06}$	$1.04_{\pm 0.03}$	$53.62_{\pm 0.57}$
	Backstroke	100.0	100.0	$0.164_{\pm 0.012}$	$0.077_{\pm 0.021}$	$0.914_{\pm 0.035}$	$0.97_{\pm 0.62}$	$1.21_{\pm 0.02}$	$54.37_{\pm 0.82}$
	Breaststroke	100.0	100.0	$0.084_{\pm 0.019}$	$0.038_{\pm 0.003}$	$0.946_{\pm 0.015}$	$0.26_{\pm 0.14}$	$1.58_{\pm 0.01}$	$89.20_{\pm 0.31}$
	Butterfly	100.0	100.0	$0.068_{\pm 0.007}$	$0.018_{\pm 0.000}$	$0.956_{\pm 0.036}$	$0.15_{\pm 0.04}$	$1.59_{\pm 0.01}$	$77.90_{\pm 0.82}$

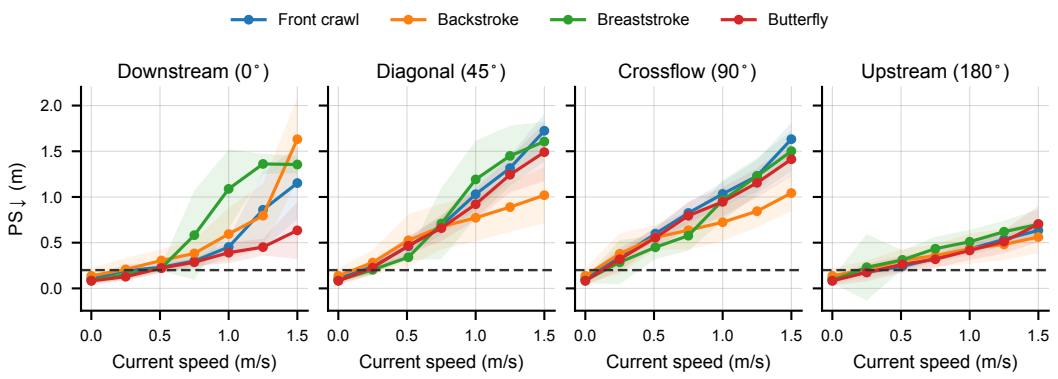

Figure 19: **Position error by flow direction.** Figure 5(b) averages the four directions; here they are separate.

B.1.5 Cross-Environment Generalization

Direction Differences In the main paper, we show performance under different flow conditions in Figure 5. Here, we analyze the effect of flow direction in more detail. Figure 19 shows that, at flow speeds from 0.25 to 1.0 m/s, downstream ($0°$) and upstream ($180°$) flows generally result in lower average PS errors than diagonal ($45°$) and crossflow ($90°$) flows. We do not analyze flow speeds above 1.0 m/s because the water surface approaches the pool boundary, sometimes causing the character to sink and resulting in large standard deviations. Overall, these results suggest that downstream and upstream flows are easier to handle than flows with a lateral component.

Various Fluids We show details of five real fluids evaluation in Table 11.

For olive oil, its lower density provides less buoyant support than water. In Figure 5(c), we showed that at densities of 500 and 750 kg/m^3, the character sinks. Nevertheless, the policy maintains an SR above 75% across all strokes, indicating that it can compensate for reduced buoyancy through its learned motion and still generate sufficient forces for floating and task completion.

Comparing seawater and heavy water, heavy water has larger LD errors (0.065, 0.207, 0.052, 0.063) than seawater (0.053, 0.138, 0.043, 0.058), while achieving higher Prog values (0.946, 1.015, 1.014, 0.975) than seawater (0.927, 0.978, 0.991, 0.968). This suggests that the higher density of heavy water allows the policy to achieve greater forward progress, but may make heading control less stable.

Comparing heavy water and glycerol, glycerol achieves lower Prog values (0.917, 0.892, 0.938, 0.932) than heavy water (0.946, 1.015, 1.014, 0.975). Honey, with substantially higher viscosity, further reduces progress. This suggests that higher viscosity increases drag and consequently reduces forward progress, while also providing stronger damping that can help suppress rapid lateral fluctuations.

Overall, within the tested range of densities and viscosities, the policy remains controllable across a range of real fluids, although performance degrades substantially for highly viscous honey. These results demonstrate the potential of our method for film and game production, where characters may need to swim in stylized or non water fluids without retraining a new controller.

Table 11: Cross 5 real fluid performance in T

Fluid	Stroke	SRV↑ (%)	SR↑ (%)	PS↓ (m)	LD↓ (m)	Prog→ 1	Roll↓ (°)	Vel↓ (rad/s)	Accel↓ (rad/s^2)
Olive oil	Front crawl	100.0	80.0	$0.153_{\pm 0.066}$	$0.048_{\pm 0.022}$	$0.837_{\pm 0.041}$	$1.37_{\pm 0.81}$	$1.08_{\pm 0.04}$	$53.86_{\pm 1.99}$
	Backstroke	100.0	76.7	$0.146_{\pm 0.087}$	$0.069_{\pm 0.064}$	$0.881_{\pm 0.059}$	$0.27_{\pm 0.32}$	$1.20_{\pm 0.02}$	$53.22_{\pm 0.97}$
	Breaststroke	100.0	100.0	$0.102_{\pm 0.038}$	$0.051_{\pm 0.035}$	$0.915_{\pm 0.039}$	$0.38_{\pm 0.47}$	$1.58_{\pm 0.05}$	$83.84_{\pm 3.94}$
	Butterfly	100.0	90.0	$0.117_{\pm 0.084}$	$0.058_{\pm 0.059}$	$0.915_{\pm 0.043}$	$0.21_{\pm 0.21}$	$1.54_{\pm 0.04}$	$75.81_{\pm 2.49}$
Seawater	Front crawl	100.0	93.3	$0.121_{\pm 0.054}$	$0.053_{\pm 0.032}$	$0.927_{\pm 0.032}$	$0.83_{\pm 0.40}$	$0.95_{\pm 0.05}$	$50.71_{\pm 1.70}$
	Backstroke	100.0	63.3	$0.200_{\pm 0.106}$	$0.138_{\pm 0.108}$	$0.978_{\pm 0.051}$	$1.19_{\pm 1.21}$	$1.15_{\pm 0.03}$	$51.03_{\pm 1.88}$
	Breaststroke	100.0	93.3	$0.137_{\pm 0.034}$	$0.043_{\pm 0.030}$	$0.991_{\pm 0.048}$	$0.45_{\pm 0.29}$	$1.57_{\pm 0.03}$	$87.00_{\pm 2.83}$
	Butterfly	100.0	86.7	$0.119_{\pm 0.075}$	$0.058_{\pm 0.058}$	$0.968_{\pm 0.026}$	$0.47_{\pm 0.38}$	$1.46_{\pm 0.05}$	$70.08_{\pm 3.40}$
Heavy water	Front crawl	96.7	86.7	$0.128_{\pm 0.053}$	$0.065_{\pm 0.040}$	$0.946_{\pm 0.030}$	$1.27_{\pm 3.67}$	$0.92_{\pm 0.04}$	$50.22_{\pm 1.66}$
	Backstroke	100.0	23.3	$0.284_{\pm 0.139}$	$0.207_{\pm 0.149}$	$1.015_{\pm 0.065}$	$2.41_{\pm 1.23}$	$1.12_{\pm 0.03}$	$49.53_{\pm 1.53}$
	Breaststroke	100.0	70.0	$0.178_{\pm 0.050}$	$0.052_{\pm 0.040}$	$1.014_{\pm 0.060}$	$0.93_{\pm 0.92}$	$1.62_{\pm 0.06}$	$93.30_{\pm 4.89}$
	Butterfly	100.0	80.0	$0.130_{\pm 0.087}$	$0.063_{\pm 0.076}$	$0.975_{\pm 0.031}$	$0.44_{\pm 0.31}$	$1.44_{\pm 0.05}$	$68.58_{\pm 3.54}$
Glycerol	Front crawl	100.0	90.0	$0.129_{\pm 0.054}$	$0.054_{\pm 0.029}$	$0.917_{\pm 0.031}$	$0.65_{\pm 0.38}$	$0.98_{\pm 0.04}$	$53.06_{\pm 2.24}$
	Backstroke	100.0	36.7	$0.369_{\pm 0.257}$	$0.229_{\pm 0.202}$	$0.892_{\pm 0.166}$	$2.95_{\pm 1.62}$	$1.17_{\pm 0.04}$	$52.50_{\pm 1.38}$
	Breaststroke	100.0	100.0	$0.083_{\pm 0.017}$	$0.037_{\pm 0.013}$	$0.938_{\pm 0.036}$	$0.66_{\pm 0.57}$	$1.69_{\pm 0.02}$	$103.42_{\pm 2.55}$
	Butterfly	100.0	96.7	$0.091_{\pm 0.034}$	$0.031_{\pm 0.020}$	$0.932_{\pm 0.025}$	$0.52_{\pm 0.48}$	$1.54_{\pm 0.05}$	$73.81_{\pm 3.09}$
Honey	Front crawl	100.0	30.0	$0.232_{\pm 0.061}$	$0.049_{\pm 0.008}$	$0.793_{\pm 0.032}$	$2.50_{\pm 0.73}$	$1.16_{\pm 0.04}$	$62.09_{\pm 2.50}$
	Backstroke	100.0	0.0	$0.749_{\pm 0.383}$	$0.276_{\pm 0.170}$	$0.548_{\pm 0.219}$	$5.11_{\pm 3.89}$	$1.23_{\pm 0.02}$	$58.44_{\pm 2.60}$
	Breaststroke	96.7	6.7	$0.361_{\pm 0.131}$	$0.048_{\pm 0.027}$	$0.736_{\pm 0.070}$	$0.71_{\pm 0.32}$	$1.80_{\pm 0.01}$	$113.66_{\pm 3.56}$
	Butterfly	100.0	60.0	$0.211_{\pm 0.071}$	$0.022_{\pm 0.007}$	$0.780_{\pm 0.037}$	$0.50_{\pm 0.36}$	$1.75_{\pm 0.02}$	$85.73_{\pm 1.43}$

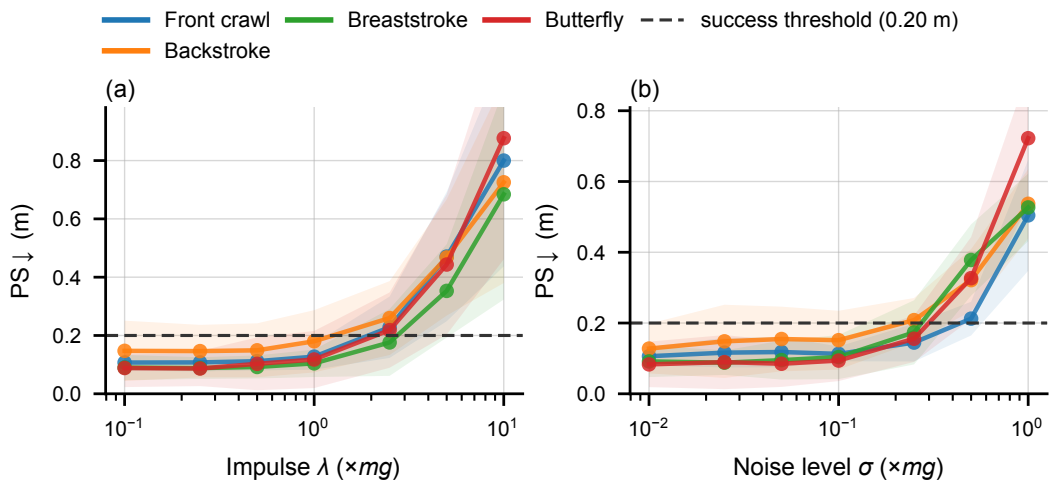

Figure 20: **Position error under perturbation.** **(a)** Position error against impulse magnitude, expressed as multiples of the character's weight mg. **(b)** continuous random force, expressed in the same units.

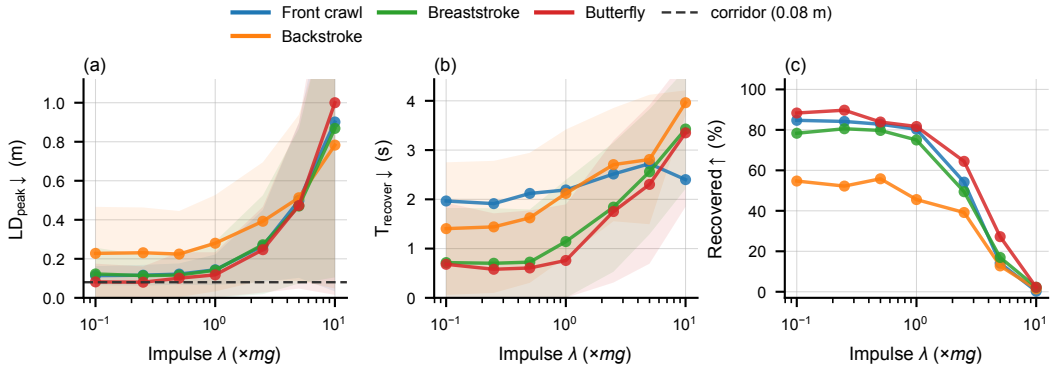

Figure 21: Recovery from impulses **(a)** $\mathrm{LD}_{\mathrm{peak}}$, the largest lateral deflection after the impulse; dashed line: nominal corridor ($\mathrm{LD} \leq 0.08\,\mathrm{m}$). **(b)** $\mathrm{T}_{\mathrm{recover}}$, the time from the impulse until the character returns to and stays within the corridor **(c)** Fraction of episodes that recover before the end of the episode.

B.1.6 Perturbation

Figure 20 and Figure 21 show that all policies remain stable under moderate perturbations and recover more than 50% toward the ideal trajectories after impulses of up to about $1\,mg$. They also remain robust under continuous random forces up to $\sigma = 0.25\,mg$.

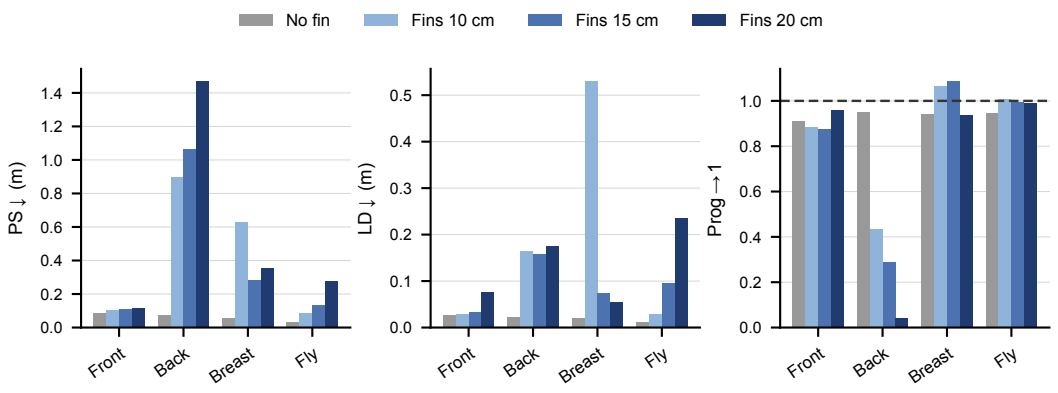

Figure 22: **Fins on a straight goal in goal-reaching task.**

B.1.7 GEOMETRIC / MORPHOLOGICAL

Figure 22 shows that, except for backstroke, fins improve propulsion and increase progress. However, fins reduce controllability: they amplify propulsion but also introduce additional drag and inertia, making it harder for the policy to match slower target speeds or steer accurately. Table 12 further confirms this limitation.

For backstroke, we find that fins provide too much buoyancy to the legs, making the upper body unstable in the water. This is mainly due to the simplified human geometry, where the upper body is positioned slightly forward relative to the legs. This simplified human geometry keeps more of the upper body in the water for other strokes, but leaves more of it above the water in backstroke, making the effect of added leg buoyancy more pronounced (Section D.2).

We surprisingly found that our policy can adapt to some cases of limb removal. For example, in Table 12, after removing a left arm or left forearm, front crawl achieves an LD error below 0.06 m. For butterfly, removing the shanks or a single leg results in an LD error below 0.09 m. Although the overall progress decreases, the supplemental videos show examples with good trajectory-following performance. However, across all limb-removal cases, the SR of all policies drops below 50%.

For each evaluation, the missing joint angles and velocities in state are filled with the corresponding reference-motion values $\mathbf{q}^{\text{ref}}$ and $\dot{\mathbf{q}}^{\text{ref}}$, while missing the link states and fluid forces of missing bodies are set to zero. These results suggest that the current environment-state representation is not geometry-invariant: removing limbs changes the structure and magnitude of the per-body force inputs, pushing the observation outside the training distribution. Designing geometry-invariant fluid-state representations is an important direction for future work.

Table 12: **The eight missing limbs and three fin sizes.** Each limb is removed from the body, or a fin of the given length is added to both feet, and the 30 base tasks are swum again at their normal speed with the policy unchanged. Only the left side of each pair is removed singly, the body being mirror-symmetric. SRV and SR are shares over the 30 tasks; the rest are mean ± sd over them.

Morphology	Stroke	SRV↑ (%)	SR↑ (%)	PS↓ (m)	LD↓ (m)	Prog→ 1	Roll↓ (°)	Vel↓ (rad/s)	Accel↓ (rad/s^2)
Left arm	Front crawl	100.0	0.0	$0.652_{\pm 0.111}$	$0.057_{\pm 0.020}$	$0.562_{\pm 0.055}$	$2.35_{\pm 0.78}$	$1.10_{\pm 0.05}$	$53.48_{\pm 2.66}$
	Backstroke	100.0	0.0	$1.147_{\pm 0.130}$	$0.185_{\pm 0.093}$	$0.286_{\pm 0.048}$	$13.88_{\pm 5.23}$	$0.97_{\pm 0.02}$	$43.45_{\pm 0.66}$
	Breaststroke	100.0	0.0	$1.130_{\pm 0.091}$	$0.105_{\pm 0.082}$	$0.240_{\pm 0.032}$	$3.67_{\pm 1.52}$	$1.21_{\pm 0.06}$	$69.08_{\pm 3.91}$
	Butterfly	100.0	0.0	$0.994_{\pm 0.208}$	$0.657_{\pm 0.181}$	$0.535_{\pm 0.098}$	$9.49_{\pm 2.76}$	$1.33_{\pm 0.03}$	$65.75_{\pm 1.64}$
Left forearm	Front crawl	100.0	0.0	$0.396_{\pm 0.085}$	$0.041_{\pm 0.018}$	$0.703_{\pm 0.045}$	$0.17_{\pm 0.29}$	$1.20_{\pm 0.04}$	$59.95_{\pm 2.09}$
	Backstroke	100.0	0.0	$0.696_{\pm 0.195}$	$0.219_{\pm 0.110}$	$0.572_{\pm 0.094}$	$3.38_{\pm 1.45}$	$1.13_{\pm 0.03}$	$48.17_{\pm 1.09}$
	Breaststroke	100.0	0.0	$0.772_{\pm 0.159}$	$0.319_{\pm 0.154}$	$0.536_{\pm 0.058}$	$9.66_{\pm 2.10}$	$1.52_{\pm 0.02}$	$86.88_{\pm 1.09}$
	Butterfly	100.0	0.0	$0.511_{\pm 0.157}$	$0.361_{\pm 0.135}$	$0.745_{\pm 0.063}$	$9.11_{\pm 1.60}$	$1.67_{\pm 0.04}$	$83.62_{\pm 2.24}$
Left leg	Front crawl	6.7	0.0	$1.513_{\pm 0.365}$	$0.298_{\pm 0.074}$	$0.108_{\pm 0.166}$	$76.23_{\pm 28.29}$	$1.29_{\pm 0.02}$	$64.23_{\pm 0.92}$
	Backstroke	80.0	0.0	$0.918_{\pm 0.268}$	$0.398_{\pm 0.228}$	$0.468_{\pm 0.124}$	$17.62_{\pm 28.40}$	$0.95_{\pm 0.02}$	$43.11_{\pm 1.25}$
	Breaststroke	96.7	0.0	$0.874_{\pm 0.151}$	$0.130_{\pm 0.088}$	$0.409_{\pm 0.084}$	$8.06_{\pm 15.59}$	$1.60_{\pm 0.05}$	$92.48_{\pm 3.88}$
	Butterfly	100.0	3.3	$0.349_{\pm 0.116}$	$0.088_{\pm 0.062}$	$0.720_{\pm 0.050}$	$2.30_{\pm 0.50}$	$1.52_{\pm 0.04}$	$71.96_{\pm 2.33}$
Left shank	Front crawl	90.0	0.0	$0.609_{\pm 0.181}$	$0.126_{\pm 0.087}$	$0.525_{\pm 0.072}$	$5.44_{\pm 3.92}$	$1.21_{\pm 0.07}$	$61.15_{\pm 4.24}$
	Backstroke	100.0	0.0	$0.913_{\pm 0.214}$	$0.545_{\pm 0.222}$	$0.527_{\pm 0.091}$	$5.29_{\pm 3.70}$	$1.06_{\pm 0.01}$	$45.18_{\pm 1.82}$
	Breaststroke	100.0	0.0	$0.479_{\pm 0.142}$	$0.097_{\pm 0.071}$	$0.648_{\pm 0.069}$	$2.82_{\pm 0.92}$	$1.62_{\pm 0.03}$	$86.92_{\pm 2.29}$
	Butterfly	100.0	50.0	$0.233_{\pm 0.114}$	$0.073_{\pm 0.063}$	$0.804_{\pm 0.049}$	$1.06_{\pm 0.76}$	$1.53_{\pm 0.03}$	$73.95_{\pm 2.00}$
Both arms	Front crawl	100.0	0.0	$1.499_{\pm 0.071}$	$0.084_{\pm 0.031}$	$0.039_{\pm 0.018}$	$2.22_{\pm 0.61}$	$0.95_{\pm 0.03}$	$42.40_{\pm 1.66}$
	Backstroke	76.7	0.0	$1.381_{\pm 0.079}$	$0.057_{\pm 0.036}$	$0.136_{\pm 0.027}$	$15.94_{\pm 26.09}$	$0.77_{\pm 0.03}$	$34.26_{\pm 1.84}$
	Breaststroke	100.0	0.0	$1.137_{\pm 0.111}$	$0.105_{\pm 0.051}$	$0.225_{\pm 0.036}$	$3.29_{\pm 3.61}$	$0.95_{\pm 0.05}$	$58.58_{\pm 2.07}$
	Butterfly	100.0	0.0	$1.062_{\pm 0.158}$	$0.126_{\pm 0.061}$	$0.304_{\pm 0.069}$	$7.54_{\pm 6.70}$	$1.11_{\pm 0.08}$	$55.37_{\pm 3.93}$
Both forearms	Front crawl	100.0	0.0	$0.859_{\pm 0.099}$	$0.113_{\pm 0.049}$	$0.413_{\pm 0.042}$	$0.88_{\pm 0.44}$	$1.03_{\pm 0.02}$	$49.39_{\pm 1.29}$
	Backstroke	100.0	0.0	$0.745_{\pm 0.133}$	$0.122_{\pm 0.084}$	$0.503_{\pm 0.049}$	$1.17_{\pm 1.58}$	$1.16_{\pm 0.03}$	$51.29_{\pm 1.30}$
	Breaststroke	100.0	0.0	$0.812_{\pm 0.110}$	$0.102_{\pm 0.060}$	$0.426_{\pm 0.040}$	$4.52_{\pm 1.62}$	$1.33_{\pm 0.01}$	$72.87_{\pm 1.21}$
	Butterfly	100.0	0.0	$0.671_{\pm 0.190}$	$0.147_{\pm 0.081}$	$0.550_{\pm 0.089}$	$1.84_{\pm 1.84}$	$1.53_{\pm 0.09}$	$77.14_{\pm 3.68}$
Both legs	Front crawl	0.0	0.0	$1.319_{\pm 0.082}$	$0.288_{\pm 0.074}$	$0.179_{\pm 0.042}$	$101.02_{\pm 4.84}$	–	–
	Backstroke	0.0	0.0	$1.262_{\pm 0.091}$	$0.133_{\pm 0.068}$	$0.229_{\pm 0.027}$	$88.98_{\pm 8.63}$	–	–
	Breaststroke	0.0	0.0	$1.387_{\pm 0.072}$	$0.073_{\pm 0.026}$	$0.115_{\pm 0.023}$	$111.64_{\pm 28.74}$	–	–
	Butterfly	100.0	0.0	$0.686_{\pm 0.147}$	$0.135_{\pm 0.080}$	$0.536_{\pm 0.042}$	$1.33_{\pm 0.41}$	$1.60_{\pm 0.03}$	$77.82_{\pm 1.91}$
Both shanks	Front crawl	0.0	0.0	$1.562_{\pm 0.075}$	$0.148_{\pm 0.011}$	$0.022_{\pm 0.040}$	$79.47_{\pm 7.28}$	–	–
	Backstroke	100.0	0.0	$0.812_{\pm 0.193}$	$0.336_{\pm 0.150}$	$0.496_{\pm 0.073}$	$9.25_{\pm 3.41}$	$1.02_{\pm 0.01}$	$46.33_{\pm 0.67}$
	Breaststroke	96.7	0.0	$1.041_{\pm 0.100}$	$0.086_{\pm 0.047}$	$0.297_{\pm 0.037}$	$8.92_{\pm 3.22}$	$1.70_{\pm 0.03}$	$88.09_{\pm 2.49}$
	Butterfly	100.0	13.3	$0.304_{\pm 0.105}$	$0.050_{\pm 0.012}$	$0.715_{\pm 0.058}$	$0.71_{\pm 0.32}$	$1.70_{\pm 0.02}$	$82.83_{\pm 1.39}$
Fins 10 cm	Front crawl	100.0	83.3	$0.153_{\pm 0.099}$	$0.056_{\pm 0.029}$	$0.863_{\pm 0.057}$	$1.26_{\pm 0.54}$	$1.04_{\pm 0.03}$	$60.16_{\pm 2.19}$
	Backstroke	13.3	0.0	$0.987_{\pm 0.285}$	$0.115_{\pm 0.059}$	$0.411_{\pm 0.137}$	$39.65_{\pm 22.03}$	$1.11_{\pm 0.05}$	$65.31_{\pm 2.70}$
	Breaststroke	70.0	6.7	$0.444_{\pm 0.329}$	$0.186_{\pm 0.135}$	$0.957_{\pm 0.249}$	$6.70_{\pm 7.78}$	$1.83_{\pm 0.06}$	$125.45_{\pm 5.89}$
	Butterfly	100.0	56.7	$0.234_{\pm 0.151}$	$0.156_{\pm 0.108}$	$0.972_{\pm 0.086}$	$0.65_{\pm 0.69}$	$1.51_{\pm 0.07}$	$73.78_{\pm 4.22}$
Fins 15 cm	Front crawl	90.0	76.7	$0.183_{\pm 0.164}$	$0.052_{\pm 0.018}$	$0.866_{\pm 0.061}$	$4.32_{\pm 14.59}$	$1.01_{\pm 0.05}$	$59.96_{\pm 2.96}$
	Backstroke	6.7	0.0	$1.083_{\pm 0.246}$	$0.251_{\pm 0.190}$	$0.339_{\pm 0.140}$	$50.39_{\pm 24.19}$	$1.07_{\pm 0.01}$	$72.08_{\pm 3.29}$
	Breaststroke	0.0	0.0	$0.527_{\pm 0.308}$	$0.100_{\pm 0.089}$	$0.832_{\pm 0.240}$	$22.65_{\pm 16.90}$	–	–
	Butterfly	100.0	33.3	$0.295_{\pm 0.147}$	$0.215_{\pm 0.119}$	$0.981_{\pm 0.044}$	$0.62_{\pm 0.45}$	$1.49_{\pm 0.05}$	$72.98_{\pm 3.12}$
Fins 20 cm	Front crawl	86.7	70.0	$0.221_{\pm 0.297}$	$0.116_{\pm 0.262}$	$0.912_{\pm 0.117}$	$4.92_{\pm 14.66}$	$1.00_{\pm 0.05}$	$60.70_{\pm 2.75}$
	Backstroke	3.3	0.0	$1.233_{\pm 0.263}$	$0.238_{\pm 0.188}$	$0.234_{\pm 0.127}$	$56.18_{\pm 26.69}$	1.07	73.09
	Breaststroke	6.7	0.0	$0.539_{\pm 0.323}$	$0.126_{\pm 0.097}$	$0.800_{\pm 0.209}$	$31.04_{\pm 27.88}$	$1.92_{\pm 0.10}$	$140.20_{\pm 15.24}$
	Butterfly	100.0	13.3	$0.344_{\pm 0.165}$	$0.266_{\pm 0.125}$	$0.991_{\pm 0.068}$	$0.69_{\pm 0.53}$	$1.50_{\pm 0.05}$	$74.23_{\pm 2.93}$

Table 13: Overall generalization task metrics: goal-reaching; trajectory-following in the 8 m swimming pool; both tasks under the E2 condition in the 5 m swimming pool; both tasks under the E1 condition in the 5 m swimming pool; both tasks under the P condition with perturbations below $1mg$ and continuous noise forces below $0.25mg$ in the 5 m swimming pool; and both tasks under the B condition in the 5 m swimming pool. Note: Since the body is mirror-symmetric, the example figures and videos also show right side removals (the right leg, lower legs, or forearm), are not included in Table 12.

Task	Stroke	SRV↑ (%)	SR↑ (%)	PS↓ (m)	LD↓ (m)	Prog→ 1	Roll↓ (°)	Vel↓ (rad/s)	Accel↓ (rad/s^2)
Cross goals	Front crawl	100.0	100.0	$0.088_{\pm 0.009}$	$0.030_{\pm 0.003}$	$0.949_{\pm 0.012}$	$0.43_{\pm 0.31}$	$1.01_{\pm 0.04}$	$52.76_{\pm 1.10}$
	Backstroke	100.0	100.0	$0.099_{\pm 0.043}$	$0.029_{\pm 0.015}$	$0.947_{\pm 0.038}$	$0.26_{\pm 0.17}$	$1.21_{\pm 0.03}$	$55.34_{\pm 0.74}$
	Breaststroke	100.0	100.0	$0.072_{\pm 0.017}$	$0.029_{\pm 0.006}$	$0.959_{\pm 0.022}$	$0.12_{\pm 0.09}$	$1.56_{\pm 0.03}$	$88.40_{\pm 1.22}$
	Butterfly	100.0	100.0	$0.054_{\pm 0.025}$	$0.017_{\pm 0.005}$	$0.961_{\pm 0.022}$	$0.14_{\pm 0.14}$	$1.57_{\pm 0.05}$	$76.59_{\pm 2.35}$
Trajectory-following	Front crawl	100.0	100.0	$0.095_{\pm 0.020}$	$0.035_{\pm 0.008}$	$0.955_{\pm 0.009}$	$0.52_{\pm 0.24}$	$1.00_{\pm 0.02}$	$52.98_{\pm 0.66}$
	Backstroke	100.0	80.0	$0.172_{\pm 0.219}$	$0.092_{\pm 0.157}$	$0.951_{\pm 0.040}$	$0.49_{\pm 0.56}$	$1.19_{\pm 0.03}$	$53.94_{\pm 1.35}$
	Breaststroke	100.0	100.0	$0.077_{\pm 0.023}$	$0.040_{\pm 0.026}$	$0.978_{\pm 0.017}$	$0.27_{\pm 0.22}$	$1.55_{\pm 0.02}$	$88.27_{\pm 1.39}$
	Butterfly	95.0	95.0	$0.063_{\pm 0.056}$	$0.026_{\pm 0.037}$	$0.971_{\pm 0.015}$	$0.46_{\pm 1.54}$	$1.55_{\pm 0.03}$	$75.57_{\pm 1.36}$
Cross fluid	Front crawl	99.3	76.0	$0.153_{\pm 0.070}$	$0.054_{\pm 0.028}$	$0.884_{\pm 0.067}$	$1.33_{\pm 1.83}$	$1.02_{\pm 0.10}$	$54.01_{\pm 4.75}$
	Backstroke	100.0	40.0	$0.350_{\pm 0.308}$	$0.184_{\pm 0.162}$	$0.863_{\pm 0.210}$	$2.39_{\pm 2.61}$	$1.17_{\pm 0.05}$	$52.94_{\pm 3.50}$
	Breaststroke	99.3	74.0	$0.172_{\pm 0.120}$	$0.046_{\pm 0.030}$	$0.919_{\pm 0.111}$	$0.63_{\pm 0.59}$	$1.65_{\pm 0.09}$	$96.13_{\pm 11.54}$
	Butterfly	100.0	82.7	$0.134_{\pm 0.083}$	$0.046_{\pm 0.053}$	$0.914_{\pm 0.078}$	$0.43_{\pm 0.37}$	$1.55_{\pm 0.12}$	$74.80_{\pm 6.70}$
Cross flows	Front crawl	100.0	25.4	$0.310_{\pm 0.154}$	$0.169_{\pm 0.140}$	$0.825_{\pm 0.082}$	$0.80_{\pm 0.62}$	$1.08_{\pm 0.06}$	$55.27_{\pm 2.04}$
	Backstroke	99.6	21.2	$0.347_{\pm 0.207}$	$0.198_{\pm 0.167}$	$0.813_{\pm 0.110}$	$1.48_{\pm 2.98}$	$1.19_{\pm 0.03}$	$53.26_{\pm 1.63}$
	Breaststroke	99.2	32.5	$0.275_{\pm 0.187}$	$0.165_{\pm 0.159}$	$0.876_{\pm 0.119}$	$1.09_{\pm 2.26}$	$1.64_{\pm 0.06}$	$93.31_{\pm 3.90}$
	Butterfly	100.0	34.6	$0.294_{\pm 0.163}$	$0.167_{\pm 0.145}$	$0.842_{\pm 0.084}$	$0.73_{\pm 0.77}$	$1.61_{\pm 0.07}$	$79.26_{\pm 4.11}$
Perturbation	Front crawl	98.8	93.4	$0.123_{\pm 0.067}$	$0.047_{\pm 0.045}$	$0.901_{\pm 0.027}$	$0.81_{\pm 1.38}$	$1.04_{\pm 0.04}$	$54.64_{\pm 2.22}$
	Backstroke	99.5	71.2	$0.166_{\pm 0.112}$	$0.081_{\pm 0.075}$	$0.895_{\pm 0.063}$	$0.95_{\pm 2.09}$	$1.21_{\pm 0.03}$	$54.84_{\pm 2.56}$
	Breaststroke	100.0	93.2	$0.108_{\pm 0.084}$	$0.055_{\pm 0.058}$	$0.934_{\pm 0.041}$	$0.48_{\pm 0.96}$	$1.58_{\pm 0.06}$	$90.69_{\pm 7.42}$
	Butterfly	99.8	87.9	$0.116_{\pm 0.138}$	$0.049_{\pm 0.092}$	$0.923_{\pm 0.051}$	$0.58_{\pm 2.46}$	$1.56_{\pm 0.05}$	$76.63_{\pm 2.90}$
Morphologies	Front crawl	70.3	20.9	$0.815_{\pm 0.567}$	$0.125_{\pm 0.124}$	$0.472_{\pm 0.336}$	$25.30_{\pm 38.97}$	$1.07_{\pm 0.10}$	$55.77_{\pm 7.01}$
	Backstroke	61.8	0.0	$1.016_{\pm 0.292}$	$0.236_{\pm 0.198}$	$0.382_{\pm 0.167}$	$27.43_{\pm 31.79}$	$1.02_{\pm 0.12}$	$45.75_{\pm 6.47}$
	Breaststroke	70.0	0.6	$0.831_{\pm 0.358}$	$0.130_{\pm 0.110}$	$0.499_{\pm 0.295}$	$19.36_{\pm 33.50}$	$1.46_{\pm 0.27}$	$83.90_{\pm 18.37}$
	Butterfly	100.0	15.5	$0.517_{\pm 0.323}$	$0.207_{\pm 0.195}$	$0.714_{\pm 0.220}$	$3.21_{\pm 4.14}$	$1.50_{\pm 0.16}$	$73.59_{\pm 7.98}$

B.1.8 OVERALL

Table 13 summarizes the overall performance across our generalization experiments.

B.2 Comparison and Ablation Study

B.2.1 Evaluation setting

As mentioned in Section A.2,we evaluate the trained policies and their generalization performance in **TxE1** across four dimensions: (1) unseen interpolated and extrapolated final target positions beyond the training task; (2) unseen trajectories with sharp turns and continuously varying curvature at interpolated and extrapolated angles beyond the training task; (3) unseen water flows, where policies trained in still water are evaluated under flows from four directions at speeds of 66.6% and 133.3% of the training target speed; and (4) unseen pool sizes, where policies trained in a small pool are evaluated in a larger pool ($5\,\mathrm{m} \times 7.5\,\mathrm{m} \times 0.6\,\mathrm{m}$) with twice the swimming distance and motion duration.

These four dimensions are combined into three evaluation conditions of increasing difficulty: (1) goal-reaching in a larger pool with still water, testing target generalization; (2) trajectory-following in a larger pool with still water, testing trajectory generalization; and (3) both tasks in a larger pool with dynamic water flows, testing combined generalization. We compute the metrics for each condition and report their average. In total, we evaluate more than 250 policies across more than 60,000 generalization experiments.

We use the **TxE1** generalization setting for this evaluation to match the full generalization protocol while keeping the computational cost manageable. Evaluating more than 250 policies under the full set of generalization conditions would be impractical.

B.2.2 Baseline

Baseline Choices. To the best of our knowledge, this is the first RL-based method for human character swimming. The most closely related work, CFC (Dou et al., 2025), models character-fluid interactions such as wading, but does not consider full-body interaction with fluids. We therefore use the closest existing methods as baselines. Specifically, we choose MimicKit-DeepMimic (Peng et al., 2018; Peng, 2025), MimicKit-AMP (Peng et al., 2021), and ADD (Zhang et al., 2025) from imitation-based human motion RL learning, and evaluate whether they can be directly transferred to our task using **our goal states, task reward, character, data and simulator**.

This setup requires only minimal adaptation to the swimming task. Without our goal state, these methods cannot determine where to move. Without our task reward, they cannot learn the task objective. Without our simulator, they cannot reproduce swimming, which requires hydrodynamic forces, whereas these methods were originally designed for land-based tasks and use simulators without fluid interactions. Without our reference motion data, they cannot perform human motion imitation. For a fair comparison, we use the same character and training settings as their original methods for all methods.

We further compare against three surrogate models to investigate the necessity of our fluid simulator for learning swimming motions: CFC (Dou et al., 2025), which uses a PINN based rigid-fluid coupling world model; MuJoCo's inertial fluid model (Vaxenburg et al., 2025; Todorov et al., 2012); and OceanSim (Song et al., 2025a) designed for underwater robots. These baselines allow us to evaluate whether existing surrogate models can provide sufficiently fast and effective learning for our task. In addition, we compare against a high resolution SPH simulator to assess the trade off between force accuracy and simulation efficiency, as high-resolution SPH can provide more accurate fluid forces but is too computationally expensive for RL training. In addition, we use a high-resolution SPH simulator as another baseline which estimates water-body forces more accurately than SWIM. Note that CFC does not release its code or dataset. We therefore follow the description in their paper to reimplement the dataset and train the corresponding PINN model.

Imitation results As in main paper discussed, Table 3 shows that our method achieves the highest SR overall. Although MimicKit-AMP and MimicKit-DeepMimic some training seed can achieve good task performance on backstroke and butterfly, their learned motions exhibit unnatural swimming patterns, with the arms flapping within a very narrow range of joint angles, resembling bird-like flapping rather than human swimming motions. Figure 23 further shows that the imitation rewards of MimicKit-AMP and MimicKit-DeepMimic are unstable. Note that the original MimicKit-DeepMimic, MimicKit-AMP, and ADD experiments report training at the scale of 100M-1B sam-

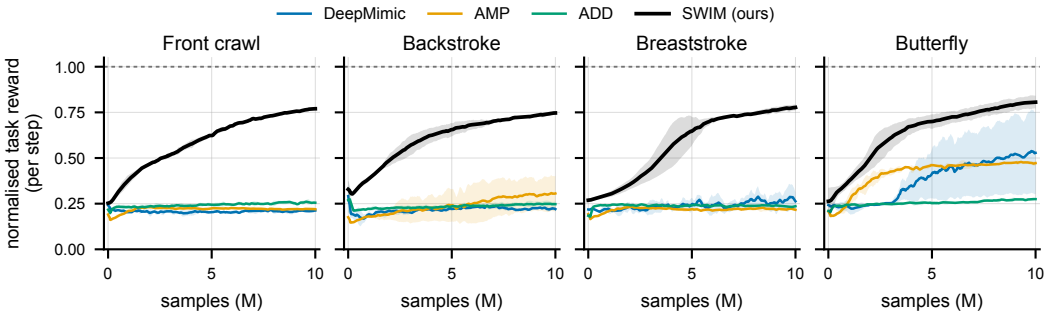

Figure 23: **Training reward of the imitation baselines.**

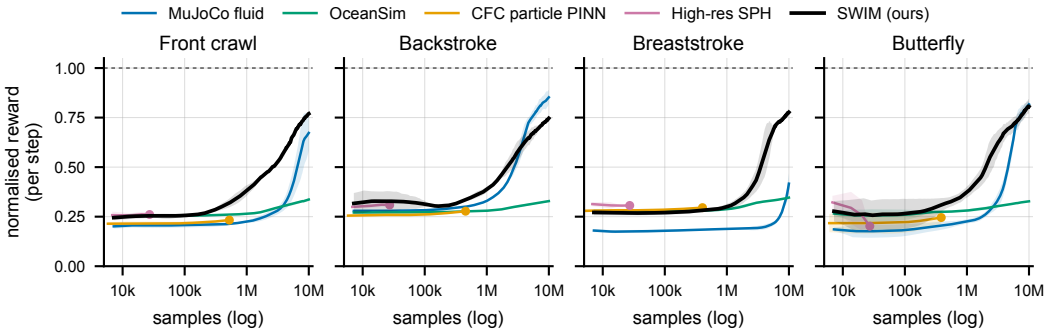

Figure 24: **Training reward of the surrogate models.**

ples scale. In contrast, for the swimming task, we cannot train for such a large number of samples due to the much higher computational cost of fluid simulation.

Overall, these results show that existing methods cannot be directly adapted to the SWIM task.

Surrogate model results Table 3 also shows that all surrogate models achieve low SRs (below 10%; training rewards in Figure 24), suggesting that inaccurate fluid forces limit their ability to learn effective swimming motions. OceanSim and MuJoCo's inertial fluid model do not model a dynamic water-air boundary and therefore cannot adequately support human swimming. In particular, swimming requires the arms to move above the water during recovery and push against the water during the stroke phase. The CFC particle-based PINN provides less accurate fluid forces and is also slower than our GPU-based SPH simulator. High-resolution SPH provides more accurate fluid forces, but is more than 200× slower than SWIM. Overall, these baselines show that SWIM provides a balance between simulation accuracy and speed that is suitable for effective RL training.

Baseline hyperparameters Table 26 shows the hyperparameters of our baseline model setting.

Table 14: **Reward, goal-state and action ablation.** As shown in Table 4: each entry is the mean over the four strokes, and ± denotes the standard deviation across them. Top-3 per column: 1st / 2nd / 3rd.

Configuration	SRV↑ (%)	SR↑ (%)	PS↓ (m)	LD↓ (m)	Prog→ 1	Roll↓ (°)	Vel↓ (rad/s)	Accel↓ (rad/s^2)
SWIM	$99.7_{\pm 0.2}$	$63.9_{\pm 4.7}$	$0.211_{\pm 0.024}$	$0.101_{\pm 0.024}$	$0.857_{\pm 0.015}$	$0.94_{\pm 0.54}$	$1.31_{\pm 0.24}$	$64.84_{\pm 14.59}$
W/O forward reward	$99.8_{\pm 0.2}$	$0.0_{\pm 0.0}$	$1.158_{\pm 0.110}$	$0.068_{\pm 0.004}$	$0.242_{\pm 0.064}$	$0.42_{\pm 0.31}$	$0.92_{\pm 0.16}$	$52.39_{\pm 14.20}$
W/O deviation reward	$99.2_{\pm 0.8}$	$57.4_{\pm 2.9}$	$0.245_{\pm 0.021}$	$0.119_{\pm 0.028}$	$0.854_{\pm 0.020}$	$0.80_{\pm 0.34}$	$1.28_{\pm 0.26}$	$65.05_{\pm 15.34}$
W/O heading reward	$99.5_{\pm 0.3}$	$67.1_{\pm 4.8}$	$0.190_{\pm 0.020}$	$0.111_{\pm 0.017}$	$0.896_{\pm 0.016}$	$0.73_{\pm 0.44}$	$1.30_{\pm 0.23}$	$63.83_{\pm 13.51}$
W/O deviation & heading rewards	$99.9_{\pm 0.1}$	$14.2_{\pm 8.5}$	$0.469_{\pm 0.086}$	$0.410_{\pm 0.082}$	$0.908_{\pm 0.016}$	$0.17_{\pm 0.08}$	$1.26_{\pm 0.29}$	$61.22_{\pm 16.69}$
W/O rolling reward	$92.7_{\pm 9.4}$	$57.5_{\pm 5.7}$	$0.233_{\pm 0.024}$	$0.103_{\pm 0.023}$	$0.849_{\pm 0.007}$	$6.92_{\pm 3.43}$	$1.30_{\pm 0.26}$	$65.80_{\pm 17.20}$
Only dense goals	$99.8_{\pm 0.2}$	$47.9_{\pm 9.4}$	$0.288_{\pm 0.041}$	$0.132_{\pm 0.062}$	$0.797_{\pm 0.027}$	$0.86_{\pm 0.17}$	$1.28_{\pm 0.23}$	$63.35_{\pm 14.52}$
Only final destination goals	$97.0_{\pm 2.8}$	$17.1_{\pm 13.9}$	$0.720_{\pm 0.418}$	$0.106_{\pm 0.014}$	$0.535_{\pm 0.267}$	$1.87_{\pm 1.29}$	$1.18_{\pm 0.19}$	$60.49_{\pm 13.18}$
W/O ω_t action	$99.7_{\pm 0.3}$	$60.8_{\pm 5.8}$	$0.215_{\pm 0.029}$	$0.099_{\pm 0.007}$	$0.850_{\pm 0.031}$	$1.00_{\pm 0.41}$	$1.13_{\pm 0.20}$	$50.39_{\pm 12.42}$

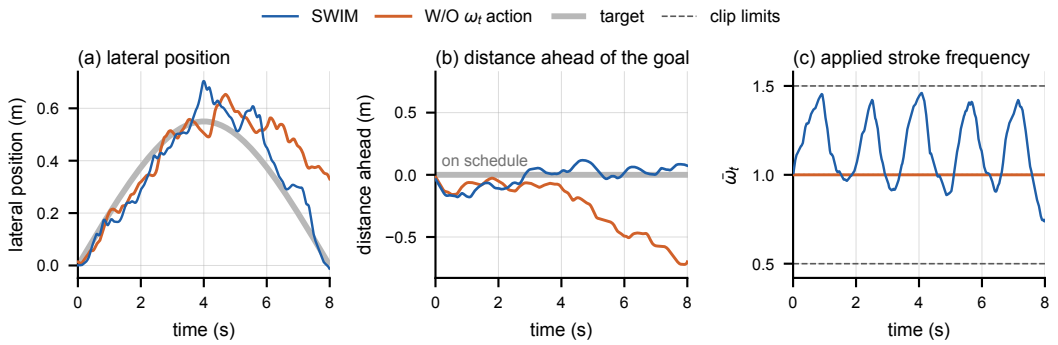

Figure 25: **Front crawl on the $A = +0.55$ m half-sine trajectory, with and without the ω_t action.** One episode per policy, shown on a common time axis. **(a)** Lateral position of the character relative to the target path. **(b)** Signed distance of the character ahead of the target intermediate goal along the forward direction of the path; 0 indicates that the character is on schedule. **(c)** Applied stroke frequency $\bar{\omega}_t$.

B.2.3 Additional Ablation Study for Goal State, Frequency Control, and Reward Design

We further show the ablation studies on multi-horizon goals, frequency control in the action space, and reward design.

First, Table 14 shows that using only dense goals or only the final destination achieves SRs of 47.9% and 17.1%, respectively, both lower than SWIM. This supports our claim that using only the final destination or dense goals is less effective for generalization.

Second, Table 14 shows that removing ω from the action space lowers the SR to 60.8%, compared with 63.9% when using ω. Figure 25 shows that, on our hardest half-sine trajectory test, the policy with ω follows the target more easily while maintaining a stable swimming speed. Interestingly, the figure shows that the policy dynamically increases and decreases its stroke frequency throughout the trajectory. Figure 26 further confirms it. Our intuition is that, because the task requires the policy to maintain an average swimming speed while following the target trajectory, frequency control allows the policy to adjust the stroke timing to maintain the desired speed. This suggests that our task and reward design may not yet be optimal, leaving room for improvement in future work.

Third, for the reward design, the results show that removing any component except the heading reward leads to a drop in generalization performance except for removing heading reward. Although removing the heading reward achieves a higher SR, Figure 27 shows that it results in larger heading errors and led to unnatural swimming motions. Note that we do not claim that our reward design is optimal for the task. Rather, these results show that each reward term contributes meaningfully to the learned swimming behavior.

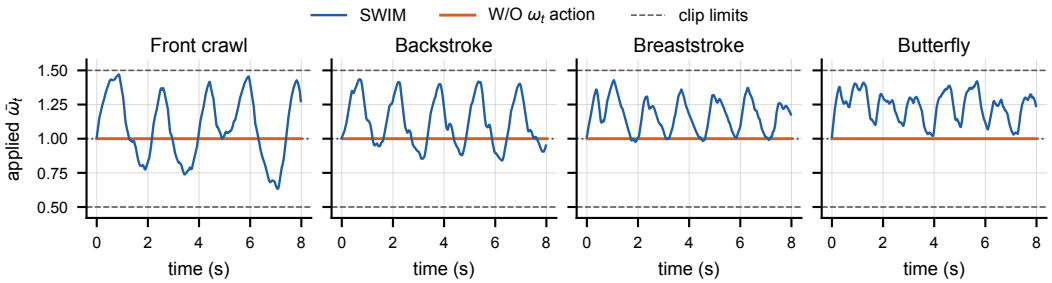

Figure 26: **Applied stroke frequency $\bar{\omega}_t$ with and without the ω_t action.**

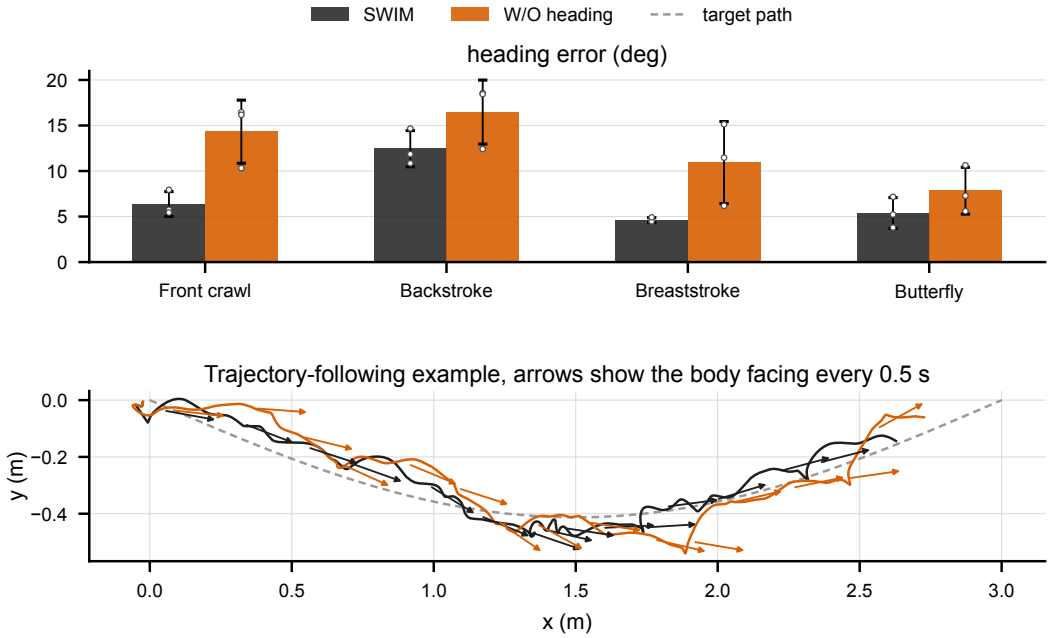

Figure 27: **W/ heading reward vs. W/O heading reward. Top:** Per-stroke mean and standard deviation over three seeds (dots), with each seed averaged over the 30 goal-reaching and trajectory-following tasks, for SWIM and the policy trained without the heading reward. Without the heading reward, the angle between the character and the path is larger. **Bottom:** One front-crawl episode viewed from above, with the body orientation shown every 0.5 s. The policy without the heading reward swims along the path with its body turned away from it.

B.3 Additional Ablation Study for the Dual Directed-Branch GCN VQ-VAE tokenizer

This section evaluates our GCN (Kipf & Welling, 2016) VQ-VAE (Van Den Oord et al., 2017) tokenizer and provides evidence for why it is an effective and robust representation of the environment state, particularly under unseen conditions.

We first summarize the key findings of our ablation study. We then introduce the architecture of our Dual Directed-Branch GCN VQ-VAE, followed by the data, evaluation metrics, experimental setup, and parameter-matching protocol. Next, we study the effects of model capacity and select the operating point through a systematic depth-width sweep. Finally, we provide additional evidence on data sufficiency, cross-gait generalization, and statistical consistency to verify that our conclusions are not specific to a particular dataset size or swimming style.

B.3.1 Key Conclusions

We first summarize the key conclusions from our architecture and ablation studies.

Table 15 shows that our Dual Directed-Branch GCN VQ-VAE tokenizer achieves the lowest NMSE on both the in-domain and OOD datasets, as well as the highest cosine similarity for reconstructed forces and torques.

Table 15: Ablation study of different VQ-VAE variants. Results are reported over three random seeds as mean ± SD. **Bold** indicates the best mean in each column.

				NMSE↓						
Model	Branch	Passing	Params	In-domain	$\text{OOD}_{\text{style}}$	$\text{OOD}_{\text{fluid}}$	OOD_{geom}	$\cos_F^{\text{limb}}$	$\mathcal{P}$	$\mathcal{U}$ %
M3 GCNN	dual	W_{sym}	10,684	0.3176 ± 0.0008	0.3924 ± 0.0005	0.3573 ± 0.0009	0.3813 ± 0.0007	0.870	398	100.0
M1 flat MLP	single	–	10,721	0.2715 ± 0.0015	0.9026 ± 0.0254	0.4316 ± 0.0038	0.4365 ± 0.0038	0.874	471	100.0
M2 flat MLP	dual	–	11,172	0.2619 ± 0.0041	0.7811 ± 0.0259	0.4106 ± 0.0086	0.4218 ± 0.0040	0.876	465	100.0
M6 GCNN	single	W_{down}	10,681	0.1992 ± 0.0001	0.3492 ± 0.0061	0.2579 ± 0.0012	0.2667 ± 0.0031	0.901	411	100.0
M5 GCNN	dual	bi	10,620	0.0254 ± 0.0006	0.0415 ± 0.0020	0.0325 ± 0.0012	0.0368 ± 0.0012	0.939	363	100.0
M4 GCNN (ours)	dual	W_{down}	10,508	$\mathbf{0.0252 \pm 0.0001}$	$\mathbf{0.0394 \pm 0.0001}$	$\mathbf{0.0310 \pm 0.0000}$	$\mathbf{0.0345 \pm 0.0000}$	**0.942**	338	100.0

Our key findings are:

First, **topology matters for OOD generalization.**

In Table 15, comparing the OOD NMSE of M1 and M2 with the GCN-based models, we find that the GCN-based models achieve lower OOD NMSE than M1 and M2, indicating that topology matters for OOD generalization. Incorporating the kinematic-tree structure substantially improves generalization to unseen swimming styles, flow conditions, and body configurations compared with topology-agnostic alternatives.

Second, **directionality matters for in-domain reconstruction.**

In Table 15, comparing the OOD NMSE of M4 and M5 with M3, we find that M4 and M5 achieve lower NMSE across all datasets than M3. This indicates that, among the GCN-based models, directionality matters for in-domain reconstruction. The proposed Dual Directed-Branch GCN propagates information along the kinematic hierarchy while avoiding unnecessary reverse message passing, leading to better reconstruction than undirected or bidirectional variants.

Third, **decoupled codebooks improve state representation.**

In Table 15, comparing the OOD NMSE of M4 and M6, we find that M4 achieves lower NMSE across all datasets than M6. This indicates that decoupled codebooks improve state representation. Separately quantizing the force and torque components provides a more effective representation of the environment state at a comparable parameter budget.

B.3.2 Proposed Architecture: Dual Directed-Branch GCN

Kinematic graph. We represent the character using a kinematic tree $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node represents one of the 13 coupled body parts and each directed edge connects a parent to its child. The root is the pelvis, and the paths from the pelvis to the limb and head extremities have depths of

from 3 to 4 edges. We therefore use three GCN layers, which allow information from the trunk to reach all body parts within one encoder pass. Although three and four layers both provide sufficient connectivity, we found that three layers perform slightly better in our experiments, so we use three layers for all models. The graph structure is derived directly from the character model and is kept fixed across all experiments.

Down-tree GCN layer. Our GCN propagates information from parent bodies to their children. For body i at layer l, the feature is updated as

$$\mathbf{h}_i^{(l+1)} = \mathbf{W}_{\text{self}}^{(l)} \mathbf{h}_i^{(l)} + \mathbf{W}_{\text{down}}^{(l)} \frac{1}{|\text{par}(i)|} \sum_{j \in \text{par}(i)} \mathbf{h}_j^{(l)}, \tag{22}$$

where the first term preserves the body's own features and the second aggregates information from its parent. The root has no parent and therefore only uses the first term. Since the kinematic structure is a tree, each non-root body has a single parent.

We compare this directed down-tree operator with two alternatives. The symmetric GCN uses both parent-to-child and child-to-parent connections with a single weight matrix. The bi-directed GCN uses separate weight matrices for the two directions. Thus, for the same feature width, the directed and bi-directed variants use more parameters than the symmetric variant. We account for this difference in our architecture comparisons.

Encoder and decoder. An encoder with depth L consists of $L-1$ hidden GCN blocks followed by a final convolution that maps the features to the latent width. Each hidden block has width H per body. The final layer outputs a latent of width k per body and per branch, which equals the codebook vector dimension d_z in Section 3.2; SWIM uses $L = 3$, $H = 32$ and $k = 3$. Each branch quantizes its latents with its own codebook of K entries, so the dual-branch tokenizer passes $2 \times 13 \times k = 26k$ values to the policy. Each hidden block contains a GCN layer, layer normalization, and GELU activation. We add residual connections from the second hidden block onward when the feature dimensions match. The final layer has no normalization, activation, or residual connection. The decoder uses the same architecture and graph connectivity in reverse. For comparison, the MLP based models use the same block structure but do not use the kinematic graph and residual connections.

B.3.3 Experimental Setup

Dataset. We train and evaluate the VQ-VAE variants using hydrodynamic interaction forces collected from our GPU DFSPH simulator at 100 Hz. We consider three spatial representations: whole-body forces and torques ($\mathbb{R}^6$), per-link forces and torques ($\mathbb{R}^{78}$), and surface-patch forces and torques ($\mathbb{R}^{312}$). The data are collected from reference-motion playback and rollouts of intermediate training policies. For each swimming stroke, we split the data by episode into training, validation, and test sets. All in-distribution data use the full-body geometry and still water.

The recorded per-link state contains a 6D force/torque vector $(F_x, F_y, F_z, T_x, T_y, T_z)$ for each of the 13 bodies. Before being used by the VQ-VAE, each wrench is transformed to the character's local frame system, compressed with $\text{sign}(x)\log(1+|x|)$, and smoothed with an exponential moving average. Thus, all reconstruction metrics are computed in the same transformed space used by the policy.

Data split. We split the data by complete episodes using an $80/10/10$ train/validation/test split. This avoids placing highly correlated consecutive frames from the same episode in different splits. The standardization statistics are computed using the training set only. We select models using validation NMSE and report test performance from the selected checkpoint.

The dataset contains 444 front-crawl episodes and 173,991 frames, collected from 144 reference-motion episodes, 297 policy-rollout episodes, and three additional evaluation episodes.

Out-of-distribution data. We evaluate generalization using three OOD splits, each changing one aspect of the training distribution. $\mathrm{OOD}_{\text{style}}$ contains three unseen swimming styles: backstroke, breaststroke, and butterfly. It contains 1,305 episodes and 522,000 frames.

$\mathrm{OOD}_{\text{fluid}}$ evaluates front crawl under disturbed-water conditions. It contains 37 episodes with six flow directions and six initial flow speeds from 0.25 to 1.5 m/s.

$\mathrm{OOD}_{\text{geom}}$ evaluates front crawl with modified body geometry. It contains 14 episodes with three fin lengths attached beyond the shanks and an intact-body condition.

Optimization. All models are trained for 60 epochs with 200 steps per epoch and a batch size of 4096. We use AdamW (Loshchilov & Hutter, 2019) with a learning rate of 3×10^{-4}, cosine annealing (Loshchilov & Hutter, 2017), and gradient norm clipping (Pascanu et al., 2013) at 1.0. Each of the two codebooks contains $K = 512$ entries. Codebook vectors are updated using exponential moving averages (Van Den Oord et al., 2017) with decay 0.99, with a commitment weight of 0.25. We initialize the codebooks from the first batch and reinitialize unused entries during training.

Metrics. We follow and report four metrics in Table 16: NMSE (Normalized Mean Squared Error), $\cos$ (Cosine Similarity), $\mathcal{P}$ (Perplexity / Effective Vocabulary Size), and $\mathcal{U}$ (Codebook Utilization). NMSE measures the reconstruction error normalized by the scale of the data. $\cos$ measures how well the reconstructed forces match the directions of the ground-truth forces. $\mathcal{P}$ measures how evenly the model uses its discrete codebook tokens, while $\mathcal{U}$ measures the percentage of available tokens that are used at least once.

Table 16: Metrics used to evaluate reconstruction quality and codebook usage.

Metric	Measures	Better
NMSE	Normalized reconstruction error	Lower
Cosine similarity	Force/torque directional agreement	Higher
Perplexity ($\mathcal{P}$)	Effective number of codes used	Higher
Code utilization ($\mathcal{U}$)	Fraction of codes used	Higher

Comparison Setup We compare six VQ-VAE architectures to isolate the effects of graph topology, message-passing direction, and dual codebooks. All models use three GCN/MLP layers and same size of codebook. For dual-branch models, the two branches use independent codebooks.

Because different graph operators use different numbers of weight matrices, matching the hidden width would result in models with different parameter counts. We therefore match the models by parameter count instead. The target is $P^* = 10{,}508$, corresponding to our proposed model (M4), and we adjust the hidden width of each architecture to approximately match this count. The resulting models are within 6.3% of the target parameter count. Codebook parameters are excluded from this count.

Table 17 lists the models. M1 and M2 are flat MLP baselines without graph structure, using single and dual branches, respectively. M3–M5 are dual-branch GCN models that compare symmetric, down-tree, and bi-directed message passing. M4 is our proposed down-tree GCN. M6 is a single-branch version of M4, which isolates the effect of using two codebooks.

Table 17: VQ-VAE architectures used in our ablation study. All models use three layers and $K = 512$ codebook entries. The hidden width is adjusted to approximately match the parameter count across models.

ID	Architecture	Branches	Width	Params	Purpose
M1	Flat MLP	Single	$(44, 22, 11)$	10,721	Unstructured baseline
M2	Flat MLP	Dual	$(36, 18, 9)$	11,172	Dual branch without graph
M3	Symmetric GCN	Dual	46	10,684	Undirected message passing
M4	Down-tree GCN	Dual	32	10,508	**Proposed model**
M5	Bi-directed GCN	Dual	26	10,620	Bidirectional message passing
M6	Down-tree GCN	Single	46	10,681	Single-branch control

B.3.4 DEPTH, WIDTH AND THE OPERATING POINT

This section explains how we choose the encoder hyperparameters.

We first study how encoder depth and width affect reconstruction and generalization. We sweep $L \in \{2, 3, 4, 5\}$ and $H \in \{4, 8, 16, 32, 64\}$ for the dual-branch down-tree GCN at $k = 3$ and $K = 512$, using three random seeds for each configuration (Table 18, Figure 28). The results support our choice of using $L = 3$ or $L = 4$, which achieve competitive performance across the tested configurations. We do not claim that this is the optimal choice; rather, the results show that it provides a good balance between reconstruction quality and generalization.

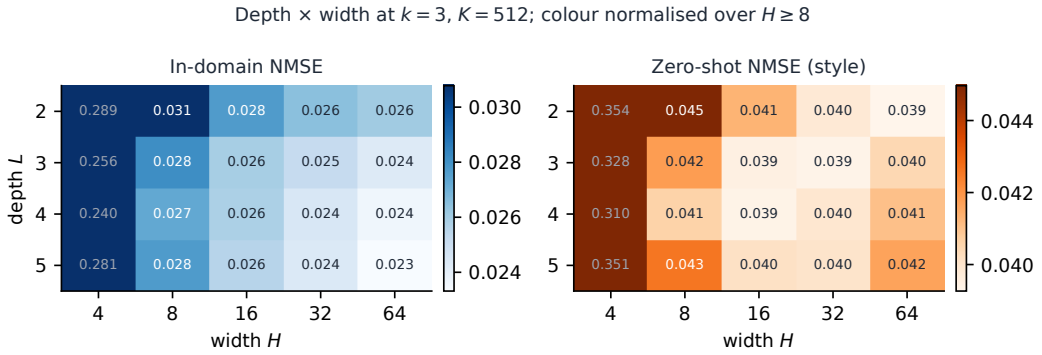

Figure 28: Depth against width at $k = 3$, $K = 512$, three seeds per cell. Color is normalized over $H \geq 8$, because the collapsed widths are an order of magnitude away and would otherwise paint the whole usable region one flat color. Above the cliff the surface is nearly flat in both axes. Reading down a column shows what depth is worth: each additional layer lowers the floor slightly and spends parameters to do it, and no depth fails the way $H{=}4$ does. Reading across a row shows the width trade, which is the one that matters, since the zero-shot panel stops improving where the in-domain panel keeps going.

B.3.5 FURTHER EVALUATION

We conduct four additional evaluations to further validate the design of our VQ-VAE: dual-codebook quantization at equal state width, training data sufficiency, cross-stroke generality, and the full parameter-matched architecture grid.

Dual Branch Vs. Single Branch A comparison at the same latent width k is not fully controlled because the dual-branch model provides twice as many dimensions to the policy: $26k$ for the dual-branch model versus $13k$ for the single-branch model. We therefore compare the two models both at equal k and at equal policy-input width. Since our grid uses $k \in \{1, 2, 3, 4\}$, the equal-width comparison matches the dual-branch model at k with the single-branch model at $2k$, giving matched state widths of 26 and 52 dimensions (Table 19).

The dual-branch model performs better in most settings. At equal k, it reduces NMSE by 2.7% for the flat MLP and 69.8% for the down-tree GCN. When the policy-input width is matched, the improvements are 6.7% and 23.2%, respectively. The same trend is observed for $\text{OOD}_{\text{style}}$. These results show that the benefit of separate force and torque codebooks is not only due to providing a larger policy state.

Training Data Sufficiency We evaluate whether the training set is sufficiently large by training our model, from 24 to all 355 episodes. The validation and test sets are kept fixed across all settings, and the standardizer is fitted only on the corresponding training subset (Table 20).

Performance improves rapidly with the first 90–180 episodes and then largely saturates. Increasing the training set from 24 to 355 episodes reduces in-domain NMSE by 8.2%, while the final increase from 180 to 355 episodes gives only a 0.5% improvement. $\text{OOD}_{\text{style}}$ shows a similar trend, improv-

Table 18: Depth L against width H for the down-tree dual-branch encoder at $k = 3$, $K = 512$, three seeds per cell, mean $\pm$ sd. **Bold** marks the best mean in each column. $\mathcal{P}$ is codebook perplexity out of $K = 512$ and $\mathcal{U}$ the fraction of entries that ever fire. Usage stays above 99.8% everywhere in this grid, including at the $H = 4$ cliff where reconstruction fails by an order of magnitude, so nothing here is a codebook collapse.

L	H	Params	NMSE	$\text{OOD}_{\text{style}}$	$\text{OOD}_{\text{fluid}}$	OOD_{geom}	$\mathcal{P}$	$\mathcal{U}\,\%$
2	4	252	0.2893 ± 0.0052	0.3540 ± 0.0045	0.3238 ± 0.0105	0.3270 ± 0.0139	408	100.0
2	8	492	0.0308 ± 0.0003	0.0450 ± 0.0009	0.0376 ± 0.0004	0.0450 ± 0.0009	351	100.0
2	16	972	0.0276 ± 0.0001	0.0413 ± 0.0002	0.0336 ± 0.0002	0.0388 ± 0.0003	339	100.0
2	32	1,932	0.0265 ± 0.0001	0.0398 ± 0.0005	0.0323 ± 0.0002	0.0364 ± 0.0001	342	100.0
2	64	3,852	0.0260 ± 0.0001	$\mathbf{0.0393 \pm 0.0005}$	0.0317 ± 0.0001	0.0355 ± 0.0002	339	100.0
3	4	428	0.2558 ± 0.0038	0.3280 ± 0.0058	0.2904 ± 0.0128	0.2910 ± 0.0120	348	100.0
3	8	1,100	0.0281 ± 0.0015	0.0418 ± 0.0024	0.0341 ± 0.0018	0.0391 ± 0.0027	346	100.0
3	16	3,212	0.0259 ± 0.0002	0.0395 ± 0.0005	0.0315 ± 0.0003	0.0354 ± 0.0002	342	100.0
3	32	10,508	0.0252 ± 0.0000	0.0393 ± 0.0002	0.0310 ± 0.0000	0.0345 ± 0.0000	338	100.0
3	64	37,388	0.0243 ± 0.0000	0.0404 ± 0.0003	0.0306 ± 0.0001	0.0338 ± 0.0001	344	99.9
4	4	604	0.2404 ± 0.0168	0.3097 ± 0.0186	0.2858 ± 0.0238	0.2853 ± 0.0220	362	99.9
4	8	1,708	0.0272 ± 0.0002	0.0406 ± 0.0002	0.0332 ± 0.0003	0.0380 ± 0.0006	338	100.0
4	16	5,452	0.0258 ± 0.0001	0.0394 ± 0.0001	0.0316 ± 0.0004	0.0352 ± 0.0004	337	100.0
4	32	19,084	0.0245 ± 0.0001	0.0399 ± 0.0001	0.0307 ± 0.0001	0.0335 ± 0.0003	338	100.0
4	64	70,924	0.0236 ± 0.0000	0.0410 ± 0.0002	0.0306 ± 0.0001	0.0333 ± 0.0001	348	100.0
5	4	780	0.2813 ± 0.0468	0.3506 ± 0.0504	0.3220 ± 0.0435	0.3360 ± 0.0475	360	100.0
5	8	2,316	0.0277 ± 0.0003	0.0426 ± 0.0007	0.0342 ± 0.0006	0.0390 ± 0.0011	345	100.0
5	16	7,692	0.0255 ± 0.0001	0.0401 ± 0.0000	0.0317 ± 0.0004	0.0352 ± 0.0003	333	100.0
5	32	27,660	0.0244 ± 0.0001	0.0400 ± 0.0002	0.0308 ± 0.0001	0.0337 ± 0.0002	338	100.0
5	64	104,460	$\mathbf{0.0233 \pm 0.0000}$	0.0416 ± 0.0003	$\mathbf{0.0306 \pm 0.0002}$	$\mathbf{0.0331 \pm 0.0001}$	345	100.0

Table 19: Dual-branch versus single-branch quantization at equal latent width k and equal policy-input width. Negative values indicate lower NMSE for the dual-branch model. Equal-k results average over $k \in \{1, 2, 3, 4\}$, while equal-width results average over the matched state widths of 26 and 52 dimensions

Pair	Control	In-domain NMSE		$\text{OOD}_{\text{style}}$	
		equal k	equal width	equal k	equal width
M2 vs M1	flat MLP	-2.7%	-6.7%	-15.1%	-16.4%
M4 vs M6	down-tree graph	-69.8%	-23.2%	-71.8%	-38.2%

ing by 11.2% from 24 to 355 episodes and by only 1.0% from 180 to 355 episodes. These results indicate that the our training set provides sufficient data for the reported results.

Table 20: Dataset size sweep.

Episodes	Transitions	Cycles	NMSE	$\text{OOD}_{\text{style}}$	$\text{OOD}_{\text{fluid}}$	OOD_{geom}
24	9,223	50	0.0279	0.0442	0.0353	0.0408
45	17,623	96	0.0270	0.0418	0.0342	0.0384
90	34,780	191	0.0261	0.0402	0.0324	0.0365
180	69,673	382	0.0257	0.0396	0.0318	0.0355
355	138,420	760	0.0256	0.0392	0.0312	0.0348

Cross-Stroke Generality The zero-shot experiments evaluate transfer from front crawl to unseen strokes. We further test whether the selected architecture is suitable across different swimming styles by refitting the same configuration on each stroke using its own training split and standardizer (Table 21). The four in-domain NMSE values are similar, ranging from 0.0230 to 0.0252, showing that the architecture performs consistently across strokes.

We also evaluate transfer in both directions for each pair of strokes. The front-crawl model transfers to backstroke, breaststroke, and butterfly with NMSEs of 0.0674, 0.0366, and 0.0368, respectively. Conversely, the corresponding stroke-specific models achieve 0.0682, 0.0423, and 0.0386 when evaluated on front crawl. This shows that the selected architecture can be trained and applied across different swimming styles, rather than being specific to front crawl.

Table 21: Cross-stroke evaluation of the selected configuration ($L = 3$, $H = 32$, $k = 3$, $10{,}508$ parameters). Each model is refitted on its own stroke using its own training split and standardizer, with three seeds. Forward transfer evaluates the front-crawl model on another stroke, while reverse transfer evaluates that stroke-specific model on front crawl.

Stroke	NMSE	$\cos_F$	front crawl $\rightarrow$ stroke	stroke $\rightarrow$ front crawl
front crawl	0.0252 ± 0.0000	0.949	–	–
backstroke	0.0230 ± 0.0000	0.959	0.0674 ± 0.0004	0.0682 ± 0.0009
breaststroke	0.0239 ± 0.0000	0.929	0.0366 ± 0.0002	0.0423 ± 0.0002
butterfly	0.0247 ± 0.0001	0.934	0.0368 ± 0.0002	0.0386 ± 0.0002

Full Parameter-Matched Grid Finally, we evaluate all six architectures across four latent widths under strict parameter matching (Table 22). For every configuration, the hidden width is adjusted to keep the total number of model parameters close to $P^* = 10{,}508$. This allows us to compare the architectures without introducing a parameter-count advantage.

The full grid shows a consistent difference between the architectures across latent widths. The down-tree GCN achieves low reconstruction error and strong OOD performance from $k = 2$ onward, while the flat MLP and symmetric GCN remain substantially higher. The single-branch down-tree model also performs worse than the dual-branch version at the same parameter budget. These results support the design choices evaluated above and show that the proposed architecture is not dependent on a single latent width.

C Full Details of Methodology

C.1 State Representation Details

C.1.1 SWIM state

Here, we describe the state $s_t = [s_t^{\text{char}}, s_t^{\text{goal}}, s_t^{\text{env}}]$. We first summarize them in in Table 23.

Character local coordinate system We use the character's local coordinate system for position and velocity features. The local frame is centered at the character root, with the x-axis aligned

Table 22: The full parameter-matched grid

Model	Passing	Branch	k	Dim	Params	NMSE	$\text{OOD}_{\text{style}}$	$\text{OOD}_{\text{fluid}}$	OOD_{geom}	$\mathcal{P}$	$\mathcal{U}$ %
M1	–	single	1	13	10,123	0.2684 ± 0.0065	0.9405 ± 0.0358	0.4325 ± 0.0101	0.4381 ± 0.0122	470	100.0
M1	–	single	2	26	10,422	0.2666 ± 0.0017	0.9286 ± 0.0268	0.4265 ± 0.0056	0.4321 ± 0.0074	452	100.0
M1	–	single	3	39	10,721	0.2715 ± 0.0015	0.9026 ± 0.0254	0.4316 ± 0.0038	0.4365 ± 0.0038	471	100.0
M1	–	single	4	52	11,020	0.2777 ± 0.0017	0.9073 ± 0.0236	0.4400 ± 0.0036	0.4448 ± 0.0057	475	100.0
M2	–	dual	1	26	10,184	$\mathbf{0.2539 \pm 0.0071}$	0.7596 ± 0.0312	0.4020 ± 0.0102	0.4092 ± 0.0117	413	99.5
M2	–	dual	2	52	10,678	0.2535 ± 0.0041	0.7748 ± 0.0195	0.3991 ± 0.0069	0.4094 ± 0.0072	458	100.0
M2	–	dual	3	78	11,172	0.2619 ± 0.0041	0.7811 ± 0.0259	0.4106 ± 0.0086	0.4218 ± 0.0040	465	100.0
M2	–	dual	4	104	10,070	0.2865 ± 0.0045	0.8075 ± 0.0096	0.4365 ± 0.0048	0.4453 ± 0.0078	454	100.0
M3	W_{sym}	dual	1	26	10,312	0.5791 ± 0.0052	0.6803 ± 0.0033	0.6277 ± 0.0074	0.6451 ± 0.0042	329	100.0
M3	W_{sym}	dual	2	52	10,498	0.4068 ± 0.0012	0.5064 ± 0.0035	0.4578 ± 0.0012	0.4767 ± 0.0012	315	99.8
M3	W_{sym}	dual	3	78	10,684	0.3176 ± 0.0008	0.3924 ± 0.0005	0.3573 ± 0.0009	0.3813 ± 0.0007	398	100.0
M3	W_{sym}	dual	4	104	10,454	0.3076 ± 0.0007	0.3845 ± 0.0009	0.3515 ± 0.0005	0.3773 ± 0.0009	439	100.0
M4	W_{down}	dual	1	26	10,568	0.2669 ± 0.0017	$\mathbf{0.3477 \pm 0.0086}$	$\mathbf{0.3070 \pm 0.0037}$	$\mathbf{0.3216 \pm 0.0028}$	273	99.2
M4	W_{down}	dual	2	52	10,250	0.0869 ± 0.0015	$\mathbf{0.1331 \pm 0.0033}$	$\mathbf{0.1127 \pm 0.0008}$	$\mathbf{0.1219 \pm 0.0017}$	373	100.0
M4	W_{down}	dual	3	78	10,508	$\mathbf{0.0252 \pm 0.0001}$	$\mathbf{0.0394 \pm 0.0001}$	$\mathbf{0.0310 \pm 0.0000}$	$\mathbf{0.0345 \pm 0.0000}$	338	100.0
M4	W_{down}	dual	4	104	10,766	$\mathbf{0.0253 \pm 0.0001}$	$\mathbf{0.0402 \pm 0.0002}$	$\mathbf{0.0315 \pm 0.0002}$	$\mathbf{0.0354 \pm 0.0002}$	348	100.0
M5	bi	dual	1	26	10,700	0.2539 ± 0.0050	0.4926 ± 0.0141	0.3445 ± 0.0047	0.3502 ± 0.0057	341	99.1
M5	bi	dual	2	52	10,306	$\mathbf{0.0817 \pm 0.0020}$	0.1889 ± 0.0080	0.1304 ± 0.0024	0.1394 ± 0.0029	357	99.9
M5	bi	dual	3	78	10,620	0.0254 ± 0.0006	0.0415 ± 0.0020	0.0325 ± 0.0012	0.0368 ± 0.0012	363	100.0
M5	bi	dual	4	104	10,214	0.0253 ± 0.0001	0.0410 ± 0.0007	0.0322 ± 0.0002	0.0372 ± 0.0003	363	99.8
M6	W_{down}	single	1	13	10,311	0.4426 ± 0.0075	0.5975 ± 0.0184	0.5049 ± 0.0115	0.5211 ± 0.0117	391	99.9
M6	W_{down}	single	2	26	10,496	0.2742 ± 0.0032	0.4615 ± 0.0113	0.3477 ± 0.0075	0.3568 ± 0.0084	432	100.0
M6	W_{down}	single	3	39	10,681	0.1992 ± 0.0001	0.3492 ± 0.0061	0.2579 ± 0.0012	0.2667 ± 0.0031	411	100.0
M6	W_{down}	single	4	52	10,450	0.1545 ± 0.0005	0.2762 ± 0.0010	0.2081 ± 0.0003	0.2182 ± 0.0009	406	100.0

Table 23: Observation composition of the SWIM state s_t.

Component	Description	Dim.
s_t^{char}: character state		174
s_t^{root}	root height, roll, linear and angular velocities	8
s_t^{link}	local positions and velocities of other bodies	72
s_t^{joint}	joint angles and velocities	72
s_t^{phase}	phase encoding	21
s_t^{freq}	applied stroke frequency ω_t	1
s_t^{goal}	4 future goals	12
s_t^{env}: environment state		84
$s_t^{\text{env,total}}$	Whole-body raw	6
$s_t^{\text{env,token}}$	Per-body token	78
Total		270

with the character's heading, the z-axis aligned with gravity, and the y-axis as the lateral direction perpendicular to both x-axis and z-axis. Let ψ_t denote the root yaw angle around the global z-axis. A global position $\mathbf{x}^{\text{glob}}$ is transformed to the local frame as:

$$\mathbf{x}^{\text{loc}} = \mathbf{R}_z(-\psi_t)\left(\mathbf{x}^{\text{glob}} - \mathbf{p}_t^{\text{root}}\right),\ \mathbf{R}_z(-\psi_t) = \begin{bmatrix} \cos\psi_t & \sin\psi_t & 0 \\ -\sin\psi_t & \cos\psi_t & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{23}$$

For linear velocities, we apply the same rotation after subtracting the root velocity.

Note that the system assumes that the character does not flip over. During training, we terminate it if the character rolls beyond 90°, which penalizes the policy for flipping over during training. As a result, the trained policy rarely flips the character.

This choice of representing everything in the local frame helps to generalize the policy, as the RL agent only needs to the goal with respect to the current local frame.

Character state In character state s_t^{char}, we use global information for root height, roll, linear and angular velocities. In addition, we use a phase state. The phase state encodes the normalized phase $\phi_t \in [0, 1)$ of the cyclic reference swimming motion. We use a multi-frequency sinusoidal encoding (Peng, 2025):

$$s_t^{\text{phase}} = [\phi_t,\ \sin(2\pi 2^0 \phi_t),\ \cos(2\pi 2^0 \phi_t),\ \ldots,\ \sin(2\pi 2^9 \phi_t),\ \cos(2\pi 2^9 \phi_t)]. \tag{24}$$

Alongside the phase, we include the stroke frequency ω_t applied at the current step (Equation (5)). This channel therefore lets the policy observe the frequency it is controlling.

Goal state The goal state is defined in Section 3.2, Figure 12 and Section A.7. In total, the goal state has 12 dimensions, consisting of four future goals.

Environment state The environment state concatenates the whole-body wrench $s_t^{\text{env,total}}$ and the per-body tokens $s_t^{\text{env,token}}$ defined in Section 3.3, a total of 84 dimensions.

We design this representation for two reasons to consider its structure and dimension. First, our ablation study shows that the two representations individually achieve the top two SR results across the four swimming strokes, although their performance varies across strokes. However, we observe that the resulting motions can be unnatural for some strokes despite their high task success. Second, concatenating the two representations substantially improves motion quality, producing more natural swimming motions across strokes, although with a modest decrease in SR. This improvement is also supported by our motion-quality metrics in the ablation study. We therefore use the combined representation as a trade-off between task performance and motion quality.

We do not claim that this is the optimal environment state in all metrics. Instead, our design is motivated by the trade-off between force-estimation accuracy and learning efficiency. Highly accurate fluid forces generally require high-resolution fluid solvers, whereas faster surrogate models can introduce errors in force estimation. SWIM therefore explores an intermediate representation that provides sufficient environmental information while remaining efficient and stable for learning. Our ablations further show that high-dimensional or unstructured environment representations are not necessarily beneficial. Within the space of structured, low-dimensional representations, the proposed combination provides a practical balance between generalization performance and motion quality.

C.2 Policy and Action Space

RL policy setup Our RL agent implementation follows (Schulman et al., 2017) uses two neural networks: a policy network $\pi_\theta(a_t \mid s_t)$ for action sampling and a value network $V_\phi(s_t)$ for state-value estimation. Both networks take the full observation vector s_t as input.

The actor and critic are implemented as two independent multilayer perceptrons without shared weights. The actor map s_t to an action a_t: $a_t = [\bar{\omega_t}, \Delta\bar{\mathbf{q}}_t] \in \mathbb{R}^{1+36}$, where $\bar{\omega_t}$ is for stroke frequency and $\Delta\bar{\mathbf{q}}_t \in \mathbb{R}^{36}$ is a correction. And the critic map s_t to a scalar value estimate $V_\phi(s_t)$. During training, actions are sampled from a diagonal Gaussian distribution,

$$a_t \sim \mathcal{N}\big(\mu_\theta(s_t), \sigma^2 I\big)\,, \tag{25}$$

Following Peng (2025); Zhang et al. (2025) where $\sigma = 0.05$ is fixed for $\Delta\bar{\mathbf{q}}_t \in \mathbb{R}^{36}$, and at evaluation time, we use the deterministic mean action $a_t = \mu_\theta(s_t)$. The $\sigma = 0.08$ is fixed for $\bar{\omega_t} \in \mathbb{R}^1$. We use the larger noise scale for $\bar{\omega_t}$ to encourage greater exploration of frequency control during training. At evaluation time, we use the deterministic mean action $a_t = \mu\theta(s_t)$.

Before being fed into the actor and critic, the observations are normalized, except for the tokenized environment state $s_t^{\text{env,token}}$.

We use Adam Kingma & Ba (2017) optimizer with learning rates 1×10^{-5} for the actor output $\Delta\bar{\mathbf{q}}_t$, 1×10^{-4} for the actor output $\bar{\omega_t}$, and 5×10^{-4} for the critic, with gradient norm clipped to 0.5 (Pascanu et al., 2013). The critic learning rate is reduced to 5×10^{-5} for the second half of training (5M to 10M samples) .

Action Smoothing and Clipping To reduce high-frequency changes in the control signals, both action channels are passed through first-order exponential moving averages (EMA) at the 100 Hz control rate, we compute:

$$\omega_t = \alpha_\omega\, \omega_{t-1} + (1 - \alpha_\omega)\, (\bar{\omega}_t), \tag{26}$$

$$\Delta\mathbf{q}_t = \alpha_q\, \Delta\mathbf{q}_{t-1} + (1 - \alpha_q)\, \Delta\bar{\mathbf{q}}_t, \tag{27}$$

where α_ω and α_q control the amount of temporal smoothing. We use $\alpha_\omega = 0.95$ and $\alpha_q = 0.85$, corresponding to time constants of approximately 0.20 s and 0.06 s. The applied ω_{t-1} is also included in the character state, allowing the policy to observe the applied frequency. Both filters are reset at the start of each episode to $\omega_0 = 1$ and $\Delta\mathbf{q}_0 = \mathbf{0}$.

Following Peng (2025), the residual action $\Delta\mathbf{q}_t \in \mathbb{R}^{36}$ is clipped to $[-\pi, \pi]$ before being added to the reference pose. The resulting target joint pose is tracked by MuJoCo's PD position controller.

The frequency control action $\omega_t \in \mathbb{R}^1$ is first clipped to $[-0.5, 0.5]$ and then shifted to $[0.5, 1.5]$. We use this range to keep the resulting stroke frequency within a range that produces plausible human swimming motions. In our experiments, frequencies from 1.0 to 1.5 produce more natural human athletic level strokes. Values above 1.5 tend to produce motions that are less consistent with human swimming.

C.3 Coupled Body–Fluid Simulation

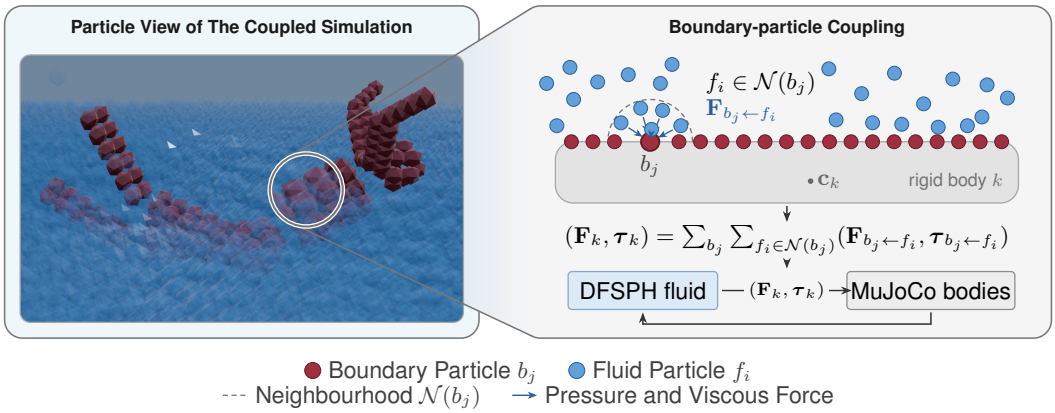

Figure 29: **Fluid and rigid-body coupling.** The coupled simulation represents the articulated character using boundary particles that interact with the fluid particles. Through boundary-particle coupling, each boundary particle b_j receives pressure and viscous forces $F_{b_j \leftarrow f_i}$ from its local fluid neighborhood $\mathcal{N}(b_j)$. Summing these contributions over all boundary particles associated with rigid body k yields the total wrench $(\mathbf{F}_k, \boldsymbol{\tau}_k)$ at its center of mass $\mathbf{c}_k$. The DFSPH fluid solver computes these forces and torques and transfers them to MuJoCo, which updates the rigid-body states.

The transition function $\mathcal{T}(s_{t+1} \mid s_t, a_t)$ modeled by a simulator that can efficiently and accurately handle dynamic water-air boundaries and rigid–fluid interactions. We therefore use the particle-based DFSPH method (Bender & Koschier, 2015) as the fluid solver, together with the rigid–fluid coupling scheme proposed by Akinci et al. (2012). We further implement the simulator on the GPU for efficient simulation. For each rigid body k, we compute the total force $\mathbf{F}_k$ and torque $\boldsymbol{\tau}_k$ by integrating the fluid forces and torques over its surface:

$$\mathbf{F}_k = \int_{\partial\mathcal{B}_k} \mathbf{f}(\mathbf{x})\, dA, \qquad \boldsymbol{\tau}_k = \int_{\partial\mathcal{B}_k} (\mathbf{x} - \mathbf{c}_k) \times \mathbf{f}(\mathbf{x})\, dA, \tag{28}$$

where $\partial\mathcal{B}_k$ is the body surface of the k-th rigid body, $\mathbf{c}_k$ is its center of mass, and $\mathbf{f}(\mathbf{x})$ denotes the fluid-induced surface force density. In practice, we form the coupling in several steps. First, we sample a set of boundary particles on each rigid body surface (Figure 29). Each boundary particle

receives the pressure and viscous forces from the surrounding fluid particles sampled within a $10\,\mathrm{cm}$ region around the body surface. Then the above integral is approximated by summing the quantities over the boundary particles:

$$\mathbf{F}_k = \sum_{b_j} \sum_{f_i \in \mathcal{N}(b_j)} \mathbf{F}_{b_j \leftarrow f_i}, \qquad \boldsymbol{\tau}_k = \sum_{b_j} \sum_{f_i \in \mathcal{N}(b_j)} \boldsymbol{\tau}_{b_j \leftarrow f_i}, \tag{29}$$

where b_j is the j-th boundary particle of the k-th rigid body, $\mathcal{N}(b_j)$ denotes its neighboring fluid particles, and $\mathbf{F}_{b_j \leftarrow f_i}$ is the force exerted by fluid particle f_i on boundary particle b_j. The torque contribution is computed as

$$\boldsymbol{\tau}_{b_j \leftarrow f_i} = \mathbf{r}_{b_j} \times \mathbf{F}_{b_j \leftarrow f_i}, \tag{30}$$

where $\mathbf{r}_{b_j} = \mathbf{x}_{b_j} - \mathbf{c}_k$ is the displacement from the center of mass of body k to boundary particle b_j.

We reimplement the fluid simulation as on GPU using NVIDIA Warp (Macklin, 2022). We thank SPlisHSPlasH, a CPU-based simulator, for providing a useful code reference (Bender et al., 2017), with DFSPH (Bender & Koschier, 2015) as the incompressible SPH fluid solver. Rigid-body dynamics and joint control are handled by MuJoCo (Todorov et al., 2012). The rigid-fluid coupling follows the same implementation used in DiffFR (Li et al., 2023), which is based on boundary-particle rigid-fluid coupling (Akinci et al., 2012).

The fluid solver runs at $200\mathrm{Hz}$ with a fixed timestep, while MuJoCo takes two fixed substeps per fluid step. The policy is queried every two fluid steps, resulting in a control frequency of $100\mathrm{Hz}$.

We use a fixed timestep instead of an adaptive CFL (Koschier et al., 2019) timestep to ensure a stable and consistent control interval for reinforcement learning. The per-body fluid forces and torques produced by the rigid–fluid coupling are used to construct the environment state. For the environment observation, we retain fluid-to-body interaction information within a near-body region defined by a $10\,\mathrm{cm}$ outward offset from the character surface.

C.4 Reward Detail

In this section, we introduce the details of reward function. As mentioned in Section 3.6:

$$r_t = r_t^{\text{task}} \cdot r_t^{\text{stab}} + w_{\text{accel}} \cdot r_t^{\text{accel}}. \tag{31}$$

$$r_t^{\text{task}} = r_t^{\text{forward}} \cdot r_t^{\text{deviation}} \cdot r_t^{\text{heading}} = \exp\left(-\alpha_p d_t^{\text{forward}}\right) \exp\left(-\alpha_a d_t^{\text{deviation}}\right) \exp\left(-\alpha_h e_t^{\text{yaw}}\right)$$

where,

$$d_t^{\text{forward}} = \left| (\mathbf{p}_t^{\text{traj}} - \mathbf{p}_t^{\text{root}}) \cdot \hat{\mathbf{d}} \right|, \tag{32}$$

$$d_t^{\text{deviation}} = \left\| \mathbf{p}_t^{\text{body}} - \text{proj}_{\mathcal{L}}(\mathbf{p}_t^{\text{body}}) \right\|_2, \tag{33}$$

$$e_t^{\text{yaw}} = \max(0, \psi_{\min}, -\psi_{\max}). \tag{34}$$

d_t^{forward}, $d_t^{\text{deviation}}$, and e_t^{yaw} are explained in Figure 4 Left and Middle. $\mathbf{p}_t^{\text{traj}}$ is the expected position on the reference trajectory at time t. $\mathbf{p}_t^{\text{root}}$ is the root position. $\hat{\mathbf{d}}$ is the direction from the initial position to the final goal. $\mathcal{L}$ is the ideal straight-line trajectory. $\text{proj}_{\mathcal{L}}(\cdot)$ projects a point onto this line. $\mathbf{p}_t^{\text{body}}$ is computed as:

$$\mathbf{p}_t^{\text{body}} = 0.5\,\mathbf{p}_t^{\text{root}} + 0.3\,\mathbf{p}_t^{\text{chest}} + 0.2\,\mathbf{p}_t^{\text{head}},$$

which encourages the central body line to stay close to the path and discourages unnatural motions where only one body part follows the trajectory.

To compute the reference trajectory at time t, we set a straight line from the initial position $\mathbf{p}^{start}$ to the final goal $\mathbf{p}^{\text{tar}}$ as the ideal trajectory during training. And the reference trajectory at time t is defined as:

$$\mathbf{p}_t^{\text{traj}} = \left(1 - \frac{t}{T}\right) \mathbf{p}^{\text{start}} + \frac{t}{T} \mathbf{p}^{\text{tar}}, \quad t \in [0, T] \tag{35}$$

where $T = 4$ second is the target motion during training.

Finally, $\psi_{\min}$ and $\psi_{\max}$ are the minimum and maximum of the four local yaw deviations ψ_i^{loc} defined in Section 3.2.

$$r_t^{\text{stab}} = \exp\left(-\alpha_r \cdot d_t^{\text{roll}}\right), \quad d_t^{\text{roll}} = \max\left(|\theta_t^{\text{roll}}| - \theta_{\text{tol}},\ 0\right). \tag{36}$$

where θ_{tol} is style-specific, informed by the roll angles of elite swimmers reported in Psycharakis & Sanders (2010).

$$r_t^{\text{accel}} = \exp\left(-\alpha_s \cdot \frac{1}{D_q} \sum_{m=1}^{D_q} |\ddot{q}_{t,m}|\right), \tag{37}$$

where $\mathbf{q}_t \in \mathbb{R}^{D_q}$ denotes the joint angles at t, $\ddot{q}_{t,m}$ is the joint acceleration, and α_s controls the penalty strength.

C.5 Learning Algorithm

Similar to prior work (Peng, 2025; Peng et al., 2018; Zhang et al., 2025), we optimize the policy using Proximal Policy Optimization (PPO) (Schulman et al., 2017). We update the policy by maximizing the clipped surrogate objective:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right], \tag{38}$$

where

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \tag{39}$$

is the PPO probability ratio, $\hat{A}_t$ is the advantage estimate, and ϵ is the clipping threshold. The critic is trained to predict the discounted return.

Table 29 lists the reward parameters used in our experiments. Each scale controls how quickly its exponential term decays with the corresponding error. With $\alpha_p = \alpha_a = 2.0$ a progress or path error of 0.1 retains $e^{-0.2} \approx 82\%$ of the term. With $\alpha_h = \alpha_r = 1.0$ the heading term, and the roll term beyond its tolerance θ_{tol}, stay above 80% for errors below 0.22 rad (12.8°). The acceleration scale was tuned in our early experiments.

Similar to (Peng et al., 2018), during training, an episode is terminated if the character flips over by more than 90° or leaves the swimming pool.

D Implementation Details

D.1 GPU-based SPH simulator

Implementation Details We reimplement the DFSPH solver (Bender & Koschier, 2015) from SPlisHSPlasH (Bender et al., 2017) on the GPU using NVIDIA Warp (Macklin, 2022). Following (Koschier et al., 2019), we use a 3D cubic-spline kernel with a uniform spatial-hash grid for efficient particle-neighbor search. The grid construction and neighbor search have linear complexity $O(n)$, where n is the number of particles. Same as (Li et al., 2023) we use a particle radius of $\Delta x = 0.025\,\text{m}$, the support radius is $h = 4\Delta x = 0.10\,\text{m}$. The hash grid is traversed once per timestep to construct a per-particle neighbor list containing up to 64 fluid and 48 boundary neighbors. The two DFSPH pressure solvers are then executed sequentially for a fixed number of iterations (4 for divergence-free and 8 for density correction), with all particle operations parallelized on the GPU. We use fixed iteration counts to keep all parallel environments in lockstep and to enable the simulation step to be captured by a CUDA Graph. Table 24 lists the core simulation parameters.

The fluid simulation operates at a fixed timestep of $5\,\text{ms}$ ($200\,\text{Hz}$). Each substep involves multiple small GPU operations, for which standard kernel dispatch introduces non-trivial CPU–GPU synchronization and kernel launch overhead. We reduce this overhead using CUDA Graphs. Because particle positions change between timesteps, dynamic operations such as rebuilding the neighbor

Table 24: Core simulation parameters.

Parameter	Value
Fluid simulator	Ours (GPU, NVIDIA Warp (Macklin, 2022))
Fluid solver	DFSPH (Bender & Koschier, 2015)
Rigid–fluid coupling	Boundary particles (Akinci et al., 2012), inset $\approx 0.5\Delta x$
Rigid-body simulator	MuJoCo (Todorov et al., 2012)
Particle radius Δx	$0.025\,\mathrm{m}$ (particle spacing $2\Delta x = 0.05\,\mathrm{m}$)
Support radius h	$4\Delta x = 0.10\,\mathrm{m}$
Rest density ρ_0	$1000\,\mathrm{kg/m^3}$
SPH timestep	$1/200\,\mathrm{s}$, fixed
MuJoCo timestep	$1/400\,\mathrm{s}$, fixed
Control frequency	$100\,\mathrm{Hz}$
Divergence solver iterations	4, fixed
Density solver iterations	8, fixed
Neighbor list capacity	64 fluid, 48 boundary
XSPH coefficient	0.05
Velocity clamp	$8\,\mathrm{m/s}$
Pool size / water depth	$3.5 \times 1.8\,\mathrm{m}$ / $0.6\,\mathrm{m}$
Parallel environments	18
Character scale	0.8
Body density	$945\,\mathrm{kg/m^3}$ (900 butterfly)
PD gains k_p / k_d	160 / 2.0 (0.5 breaststroke)

search grid are performed separately before replaying the graph. We additionally apply XSPH (Bender et al., 2017) velocity smoothing with a coefficient of 0.05 and clamp particle velocities to $8\,\mathrm{m/s}$ as a safety measure.

For RL training, multiple environments are simulated concurrently by storing their particles in shared GPU arrays and using a shared spatial-hash grid. Each environment is placed at a separate spatial offset to prevent particles from different environments from interacting.

Boundary inset calibration. The standard particle boundary (Akinci et al., 2012) leaves a fluid-free shell of approximately $0.5\Delta x$ outside each rigid surface. At the high resolutions typically used in computer graphics simulation, this shell is negligible. In our RL setting, however, the character is scaled down and simulated at a coarse particle spacing, so for the thinnest limbs, the shell increases the effective displaced volume by $2\times$ to $3.5\times$. This introduces incorrect buoyancy that biases policy learning. With a real mass, the character floats at $\Delta x = 0.05\,\mathrm{m}$ but sinks at finer resolutions.

We correct this without runtime cost by moving the boundary particles approximately $0.5\Delta x$ inside the surface, following the boundary placement of mDBC (English et al., 2022). We calibrate the inset on the primitive shape (sphere, capsule, cylinder or box) so that its measured buoyant force matches $\rho g V$.

Verification We validate the coupled simulator with quantitative checks across static, dynamic, and violent regimes. In a static fluid, the simulated pressure matches the analytical hydrostatic pressure profile, with a coefficient of determination of $R^2 = 0.98$ and a density error of 0.06%. Submerged primitives reproduce the Archimedean buoyancy force within 2%. Blocks with densities of 500, 1000, and $1500\,\mathrm{kg/m^3}$ float, remain suspended, and sink as expected, including in a sloshing pool. A towed sphere experiences a drag force with the expected direction and magnitude. In a dam-break test, the peak density error remains below 1.9%. We also evaluate the visual results for randomly controlled joint angles and for multiple characters with different densities in a single swimming pool, under both static and dynamic water conditions. In all cases, the resulting motion and fluid–character interactions appear physically plausible.

Viscosity. For water, we follow SPlisHSPlasH (Bender et al., 2017) and use only XSPH velocity smoothing (coefficient 0.05) to model effective viscosity. For the viscous fluids in our generalization tests (olive oil, glycerol, and honey, as well as seawater and heavy water), we instead model physical

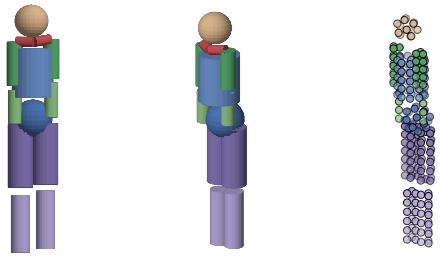

Figure 30: Character geometry.

Table 25: Character links.

Link	Shape	Radius (m)	Length (m)
Pelvis	sphere	0.088	–
Thorax	cylinder	0.094	0.236
Head	sphere	0.076	–
Clavicle	cylinder	0.021	0.082
Upper arm	cylinder	0.034	0.198
Forearm	cylinder	0.029	0.161
Thigh	cylinder	0.056	0.298
Shank	cylinder	0.042	0.278

viscosity using the implicit solver of Weiler et al. (2018), with 10 fixed iterations per step, and disable XSPH.

Cache. To avoid unnecessary computation and simulator initialization instability, we run each simulation for 0.2 seconds with the character fixed and save the resulting state as a cache. During training and evaluation, we directly load the cached state.

D.2 Humanoid Character

We use a simplified articulated humanoid with a 6-DoF floating root and 36 actuated hinge DoFs (Figure 30), corresponding to the reference data described in Section A.2. The root is not directly actuated by the policy; instead, the policy controls the joint torques, and the character moves through articulated-body dynamics and body–fluid interactions. The character has 13 rigid links: pelvis, thorax, head, and left/right clavicles, upper arms, forearms, thighs, and shanks. Each of the 12 non-root joints is modeled as three orthogonal hinges.

The pelvis and head are represented as spheres, while the other links are represented as cylinders (Table 25). The link sizes are fitted to the motion-capture subject (LABIOMEP & Qualisys, 2025) and scaled by 0.8. We use a body density of $945\,\mathrm{kg/m^3}$ for all strokes except butterfly, for which we use $900\,\mathrm{kg/m^3}$. This adjustment is based on our observation that the arms experience self-collisions during butterfly swimming, which can otherwise lead to an excessively low effective density when self-collision handling is disabled. We therefore manually reduce the butterfly body density to $900\,\mathrm{kg/m^3}$.

SPH naturally supports arbitrary geometries, and a more anatomically accurate human geometry could therefore be used in future work.

The total mass is approximately $21.4\,\mathrm{kg}$. Self-collision is disabled, so the links interact with each other only through the fluid. At $\Delta x = 0.025\,\mathrm{m}$, the character is represented by 209 boundary particles.

Table 26: **Baseline settings.** LR denotes the actor/critic learning rates, along with the discriminator learning rate for MimicKit-AMP and ADD. All methods use $4\,\mathrm{s}$ training episodes at $100\,\mathrm{Hz}$. Samples/s denotes the training throughput on a single RTX 5090, including policy updates; for Data Replay, which does not require training, it denotes the simulation rate. † indicates that the method was trained for the same wall-clock time as SWIM rather than for 10M samples. High-resolution SPH supports a maximum of 17 environments due to GPU memory limitations.

	Setup		Optimization		Training		
Method	Simulator	Reward	Optimizer	LR	Parallel envs	Samples	Samples/s
SWIM (ours)	Ours (DFSPH)	Task	Adam	1×10^{-5} / 5×10^{-4}	18	10M	1,023
MimicKit-AMP	Ours (DFSPH)	Style : task $= 0.5 : 0.5$	SGD, mom. 0.9	5×10^{-5} / 1×10^{-4} / 2.5×10^{-4}	18	10M	940
MimicKit-DeepMimic	Ours (DFSPH)	Imitation : task $= 0.7 : 0.3$	SGD, mom. 0.9	1×10^{-4} / 1×10^{-4}	18	10M	903
ADD	Ours (DFSPH)	Style : task $= 0.5 : 0.5$	SGD, mom. 0.9	1×10^{-4} / 1×10^{-4} / 2.5×10^{-4}	18	10M	899
Data Replay	Ours (DFSPH)	–	–	–	18	–	1,160
MuJoCo	MuJoCo fluid model	Task	Adam	1×10^{-5} / 5×10^{-4}	18	10M	2,194
CFC†	Particle PINN	Task	Adam	1×10^{-5} / 5×10^{-4}	18	0.39M–0.57M	55
OceanSim	Velocity damping	Task	Adam	1×10^{-5} / 5×10^{-4}	18	10M	2,657
High-res SPH†	Ours, $\Delta x = 0.0125\,\mathrm{m}$	Task	Adam	1×10^{-5} / 5×10^{-4}	17	13.6k–27.2k	5

D.3 Other Training and Evaluation Configurations

During evaluation, unseen conditions can push some inputs outside the range observed during training. We therefore clip each state dimension to its 0.375th–99.625th percentile range, computed from 200k observations sampled from the last 25% of training. Clipping is applied before normalization. We do not clip the VQ tokens, as they come from a fixed codebook and are always within the range seen during training.

All environment-state forces and torques are expressed in the character's heading frame and compressed using $\mathrm{sign}(x)\log(1+|x|)$. For the per-frame VQ-VAE inputs, we further smooth the compressed signals with an exponential moving average (EMA), using $\alpha = 0.22$ at $100\,\mathrm{Hz}$, corresponding to a time constant of approximately $0.04\,\mathrm{s}$. This matches the preprocessing used to train the codebook. The raw force and torque observations use the unsmoothed signals.

For the body-surface patches, we divide each of the 13 bodies into 4 equal-length segments along its main axis and sum the forces and torques on the boundary particles of each segment, resulting in 52 patches. For the high-dimensional temporal state, the ST-VQVAE encodes a $1.5\,\mathrm{s}$ window (150 frames) of the unsmoothed, compressed signals.

During PPO training, we stop the policy update early when the approximate KL divergence between the old and new policies exceeds 0.02, following the CleanRL implementation (Huang et al., 2022). In our early experiments, early stopping keeps the PPO clip fraction around 20% (at most 37%), compared with about 80% without early stopping, and is a key factor in stabilizing training.

For the 'ICLR' trajectory-following demo shown in Figure 7, we further trained the backstroke and breaststroke policies for an additional 5M samples to improve performance on the more challenging tracking task. All other policies and experiments were trained for 10M samples.

Computing Cost. All timings are measured on a single workstation equipped with an AMD Ryzen Threadripper 9960X CPU (24 cores, 48 threads), 128 GB of RAM, and one NVIDIA GeForce RTX 5090 GPU (32 GB). All experiments in Table A.1, including training the 276 policies and evaluating 82,872 experiments, take approximately 830 GPU-hours on a single NVIDIA RTX 5090.

Hyperparameters. Table 27 summarizes the network, action-space, and optimization settings of SWIM. Table 26 summarizes the settings of the baselines. Table 28 summarizes the PPO training parameters. Table 29 shows our reward parameter settings.

Table 27: Policy and value network configuration.

Item	Value
Input dimension	$\lvert s_t \rvert = 270$
Actor MLP	$\lvert s_t \rvert \to 512 \to 256 \to (36 + 1)$
Critic MLP	$\lvert s_t \rvert \to 512 \to 256 \to 1$
Hidden activation	ReLU6
Action space	$a_t = [\omega_t, \Delta\mathbf{q}_t] \in \mathbb{R}^{1+36}$
Action distribution	Diagonal Gaussian, $\sigma_\omega = 0.08$, $\sigma_q = 0.05$
Evaluation action	Deterministic mean
Action clip	$\omega_t \in [0.5, 1.5]$, $\Delta\mathbf{q}_t \in [-\pi, \pi]$
Action smoothing (EMA)	$\alpha_\omega = 0.95$, $\alpha_q = 0.85$
Controller	MuJoCo PD position control

Table 28: PPO training parameters.

Parameter	Value
Sample budget	10M environment samples
Discount factor γ	0.99
GAE λ	0.95
PPO clip ϵ	0.2
Value loss weight	0.5
Mini-batch size	2048
PPO epochs per update	10, early stop at KL 0.02
New episodes per epoch	18
Rollout buffer	18 episodes (on-policy)
Training episode length	4 s
Policy frequency	100 Hz
Optimizer	Adam, gradient norm clip 0.5
Actor LR	1×10^{-5}
Cadence head LR	1×10^{-4}
Critic LR	5×10^{-4}, then 5×10^{-5} after 5M samples

Table 29: Reward parameters used in our experiments.

Group	Symbol	Value
Forward progress	α_p	2.0
Path deviation	α_a	2.0
Heading	α_h	1.0
Roll stability	α_r	1.0
Joint acceleration	α_s	0.005
Acceleration weight	w_{accel}	0.05
Front crawl, backstroke roll tolerance	θ_{tol}	30°
Breaststroke, butterfly roll tolerance	θ_{tol}	10°