Learning to Ball: Composing Policies for Long-Horizon Basketball Moves

PEI XU, Stanford University, USA
ZHEN WU, Stanford University, USA
RUOCHENG WANG, Stanford University, USA
VISHNU SARUKKAI, Stanford University, USA
KAYVON FATAHALIAN, Stanford University, USA
IOANNIS KARAMOUZAS, University of California, Riverside, USA
VICTOR ZORDAN, Roblox, USA and Clemson University, USA
C. KAREN LIU, Stanford University, USA

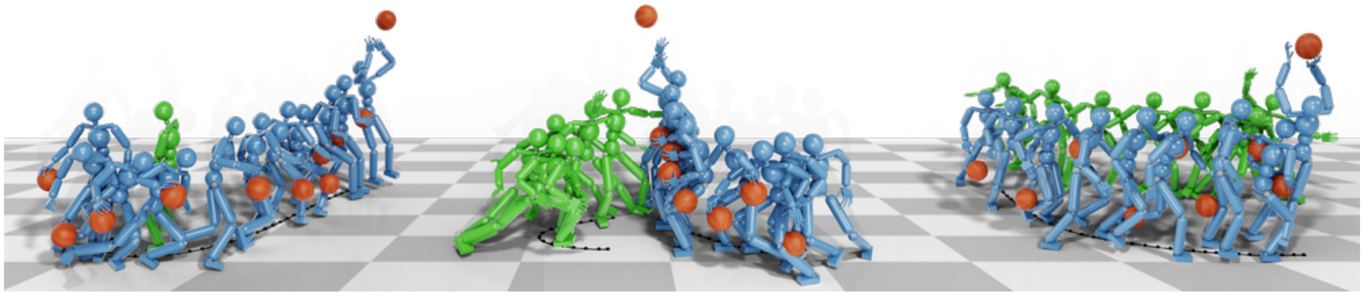


Fig. 1. We introduce a novel policy integration framework to enable the composition of drastically different motor skills in multi-phase, long-horizon tasks, among them, shoot-off-the-dribble, catch-and-shoot, and board-and-bang (grabbing an offensive rebound and scoring immediately).

Learning a control policy for a multi-phase, long-horizon task, such as basketball maneuvers, remains challenging for reinforcement learning approaches due to the need for seamless policy composition and transitions between skills. A long-horizon task typically consists of distinct subtasks with well-defined goals, separated by transitional subtasks with unclear goals but critical to the success of the entire task. Existing methods like the mixture of experts and skill chaining struggle with tasks where individual policies do not share significant commonly explored states or lack well-defined initial and terminal states between different phases. In this paper, we introduce a novel policy integration framework to enable the composition of drastically different motor skills in multi-phase long-horizon tasks with ill-defined intermediate states. Based on that, we further introduce a high-level soft router to enable seamless and robust transitions between the subtasks. We evaluate our framework on a set of fundamental basketball skills and challenging transitions. Policies trained by our approach can effectively control the simulated character to interact with the ball and accomplish the long-horizon task specified by real-time user commands, without relying on ball trajectory references.

Authors' Contact Information: Pei Xu, Stanford University, USA, peixu@stanford.edu; Zhen Wu, Stanford University, USA; Ruocheng Wang, Stanford University, USA; Vishnu Sarukkai, Stanford University, USA; Kayvon Fatahalian, Stanford University, USA; Ioannis Karamouzas, University of California, Riverside, USA; Victor Zordan, Roblox, USA and Clemson University, USA; C. Karen Liu, Stanford University, USA.

ACM 1557-7368/2025/12-ART233
https://doi.org/10.1145/3763367

CCS Concepts: • **Computing methodologies** → **Animation**; *Physical simulation*; *Reinforcement learning*.

Additional Key Words and Phrases: character animation, physics-based control, motion synthesis, hierarchical reinforcement learning

ACM Reference Format:
Pei Xu, Zhen Wu, Ruocheng Wang, Vishnu Sarukkai, Kayvon Fatahalian, Ioannis Karamouzas, Victor Zordan, and C. Karen Liu. 2025. Learning to Ball: Composing Policies for Long-Horizon Basketball Moves. *ACM Trans. Graph.* 44, 6, Article 233 (December 2025), 23 pages. https://doi.org/10.1145/3763367

1 Introduction

Many real-world tasks consist of complex objectives that can be broken down into sequences of differing subtasks. Successfully executing these multi-phase, long-horizon tasks demands the mastery of heterogeneous skills and the ability to transition seamlessly between them. Basketball provides a compelling example of these challenges. For example, a fundamental maneuver, "shoot-off-the-dribble", involves distinct subtasks such as dribbling, gathering the ball and shooting, as well as the ability to transition between these skills, ultimately culminating in the ball successfully going into the hoop. However, while *dribble* and *shoot* are characterized by well-defined stand-alone goals, *gather* acts largely as a transition subtask with poorly defined starting and ending states. Thus, executing such multi-phase tasks challenges control methods proposed to date.

Reinforcement learning (RL) has shown promise in training policies for individual skills for physics-based character control [Chentanez et al. 2018; Kwiatkowski et al. 2022; Peng et al. 2018a; Shi et al. 2023; Yin et al. 2021], but composing these policies into a cohesive framework remains an open problem. Previous approaches have attempted to address this through methods like a mixture of experts [Peng et al. 2018b; Won et al. 2020]. However, this technique relies on sufficiently exploring shared states across individual policies—a condition that does not hold for tasks like dribbling and shooting. Another line of work known as skill chaining [Chen et al. 2023a; Clegg et al. 2018; Konidaris and Barto 2009; Lee et al. 2021; Liu and Hodgins 2017] allows concatenation of policies but requires each skill to have a well-defined set of terminal states. This limitation renders it ineffective for intermediate tasks where the subtask's goal depends on the context of the subsequent policy. For example, the gathering motion between dribbling and shooting can be intuitively described as "bringing the agent to a state where shooting is possible." However, crafting a reward function based solely on state or action variables to reflect this goal is challenging.

To tackle the problem of building policies for ill-defined, intermediate subtasks, we introduce a policy integration method to compose drastically different and/or ill-defined skills to achieve a multi-phase, long-horizon task. The core idea is to first train policies for well-defined subtasks independently and then use these policies to guide the training of the ambiguous, intermediate subtasks. Specifically, for a task sequence consisting of subtasks A, B, and C, where A and C have well-defined task goals but B does not, we train B using policy A to define a valid initial state distribution and policy C's state value function to shape the terminal reward. To further improve the transition between B and C, we simultaneously adapt the pretrained policy C to the states generated by B under training. In this process, a state value estimator optimized in tandem with the adapted C will be provided to reflect the up-to-date state value evaluation for policy B optimization. With the primitive policies for all subtasks in place, we finally train a high-level soft-routing policy that directs the execution of those primitive policies based on real-time external commands, such as dribbling destination and velocity, or a jump-shot action.

Another challenging aspect in learning policies for multi-phase, long-horizon tasks is the heterogeneity of movement that demands diverse and extensive human motion data. Previous work on basketball motion synthesis has demonstrated compelling results when using structured data with corresponding full body, fingers, and basketball movements for physics-based character control [Liu and Hodgins 2018; Starke et al. 2020; Wang et al. 2023, 2024e]. However, such a special dataset is hard to scale for training a general policy capable of performing under a wide array of conditions. In our work, instead, we demonstrate the generation of policies from unstructured data. We leverage a diverse collection of basketball motion data, including full-body motions without hands, and hand-only motions, as well as motion examples from unstructured videos. To enrich locomotion behaviors, some normal running motions are also included. Our method makes no assumptions about the correspondence across datasets or availability of ball trajectories.

Our results show that the proposed method enables the agent to perform smooth and coordinated basketball maneuvers, from gross body movements to fine finger actions, while responding adaptively to user commands. The agent can freely play basketball in real-time—for instance, dribbling to any location at variable speeds and finishing with a jump shot from any direction, achieving a shooting accuracy of 91.8% on a professional court. We further demonstrate team play with multiple agents interacting through catching, passing, rebounding, and defending. Extensive ablation studies validate key design choices, such as soft routing and policy fine-tuning, and expose the limitations of existing methods in handling ambiguous subtasks. By addressing skill integration and phase transitions in long-horizon tasks, our approach advances the capabilities of reinforcement learning in dynamic, interactive environments.

2 Related Work

Creating robust controllers for long-horizon, multi-phase tasks remains a core challenge in reinforcement learning (RL) and character animation. Our work mainly builds on two areas of research: deep RL for physics-based character control and policy composition for executing multi-phase tasks.

Traditional approaches of physics-based character control, including dexterous control, typically rely on trajectory optimization and/or manually designed heuristic rules to achieve control in a planning or classic optimization way [Chen et al. 2023b; Liu 2008, 2009; Mordatch et al. 2012; Wang et al. 2013; Ye and Liu 2012]. Early work also explored using pre-collected mocap [Kry and Pai 2006; Pollard and Zordan 2005; Zhao et al. 2013] to generate human-like motions through imitation. During recent years, imitation learning using deep RL for policy optimization has drawn wide attention and become a popular approach for physics-based character control [Ling et al. 2020; Merel et al. 2017; Peng et al. 2018a, 2022, 2021; Won et al. 2020; Xu and Karamouzas 2021; Xu et al. 2023a; Yao et al. 2022, 2023; Zhu et al. 2023]. The deep RL framework has demonstrated remarkable success in training policies for physics-based character control across diverse domains, including locomotion [Peng et al. 2017; Xie et al. 2020], racket sports [Wang et al. 2024a; Zhang et al. 2023], , ball games [Kim et al. 2025; Liu and Hodgins 2018; Liu et al. 2022; Wang et al. 2024e], instrument performance [Luo et al. 2024; Wang et al. 2024d; Xu and Wang 2024; Zakka et al. 2023] and object manipulation [Bae et al. 2023; Wu et al. 2024; Xie et al. 2023; Yang et al. 2022]. Our approach follows the recent paradigm of adversarial imitation learning [Peng et al. 2021; Xu and Karamouzas 2021] combining a GAN-like architecture with reinforcement learning for motion imitation with goal-directed control, and perform motion synthesis given partially observable reference motions collected from multiple disparate sources. Policies trained with our approach enable the character to interact with the basketball validly in a human-like manner, without ball trajectories references.

Long-horizon strategic behaviors in real life, like basketball playing, often involve multiple-phased subtasks, and demand the executor to effectively chain a bunch of distinct primitive skills into a coherent sequence for task execution. To chain multiple primitive policies, traditional methods often assume that two consecutive skills will share some common states in which the succeeding policy can take over the character [Liu and Hodgins 2017; Pan et al. 2024; Wang et al. 2024b,e; Xiao et al. 2023; Xu and Karamouzas 2021]. Recent research focuses on aligning skill transitions through state distribution matching. Techniques include regularizing terminal distributions [Lee et al. 2021], modifying initial state distributions [Konidaris and Barto 2009], or employing bi-directional optimization to iteratively refine both [Chen et al. 2023a]. While these methods improve robustness, they struggle when state distributions between skills are ill-defined. For example, in the shoot-off-the-dribble task, dribbling and shooting are subtasks with clearly defined goals, but may have completely disjoint in-betweening states. The intermediate ball-gather behaviors are needed to close the gap. However, there is no clear phase division for dribbling and shooting, and, thereby, we cannot train a ball-gather policy simply in a standalone way. In this work, we present an approach to achieve such intermediate policies with ill-defined initial and terminal states. Based on that, a high-level policy is introduced to perform policy composition hierarchically for seamless transitions between primitive policies.

Previous literature explored hierarchical reinforcement learning for planning tasks [Kulkarni et al. 2016; Peng et al. 2017; Vezhnevets et al. 2017], where high-level controllers generate goals for low-level controllers to execute, while in this work, we focus on composing multiple policies in a mixture-of-experts style through a hierarchical architecture. Early work uses hierarchical architectures to compose multiple primitive policies/poses through weighted averaging in single-phase tasks [Peng et al. 2019; Ranganath et al. 2019], or through hard routing [Tessler et al. 2017; Wang et al. 2024a,e] to pick a primitive policy from a pre-trained policy set for control at each moment. To generate human-like motions, the former approach typically needs additional imitation learning during composition to avoid unnatural behaviors caused by too much averaging. The latter one, on the other hand, would suffer the challenge of policy transition when switching between different primitive policies. Our proposed soft-routing scheme allows policy composition by weighted averaging, and, meanwhile, encourages one primitive policy to dominate the control at each moment, thereby ensuring seamless and natural transition between heterogeneous policies.

3 Method Overview

We present a novel method for composing RL policies across distinct subtasks, enabling a physically simulated character to perform long-horizon, complex tasks such as playing basketball. A proficient basketball player must execute a wide range of fundamental skills and, importantly, transition fluidly between them without losing control of the ball. While the complete repertoire of basketball skills is extensive, we select a core set of seven skills and their transitions to showcase our method's ability to generate physically simulated players capable of unscripted, continuous, and interactive basketball behaviors (Figure 2). The main focus of this work is to enable seamless transitions between subtasks. We categorize these transitions into three types, ordered by increasing difficulty. This hierarchy also guides our approach when selecting the appropriate transition method for a given pair of policies:

A. **Direct Execution**: Succeeding policy can be executed directly from terminal state of the preceding policy. Used when transitioning between behaviors when the two consecutive policies share common states generally, such as *Shoot* to *Locomotion* or *Locomotion* to *Defend*.
B. **Mutual Adaptation**: Succeeding policy must adapt to novel initial states produced by the preceding policy while the preceding policy must lead to a state manageable by the succeeding policy. For instance, transitioning from *Catch* to *Shoot* may fail under direct execution if the ball is not in a familiar configuration for the shooting policy.
C. **Intermediate Policy**: This transition requires an intermediate policy to bridge incompatible subtasks. For example, transitioning from *Dribble* to *Shoot* or *Pass* often demands a *Gather* policy to reposition the ball appropriately.

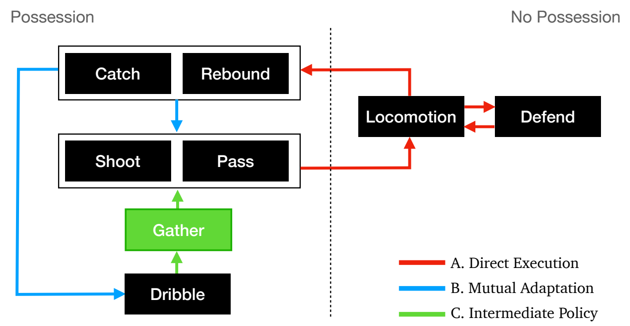


Fig. 2. Our physically simulated character is capable of performing seven distinctive basketball skills and transitioning between them. The policies (shown as black boxes) include both offensive and defensive skills. The transitions between subtasks can be categorized into three types ordered by increasing difficulty: A. **Direct Execution**; B. **Mutual Adaptation**; and C. **Intermediate Policy**. The most challenging case, Type C, requires an additional intermediate policy (shown as a green box) to be trained to bridge the gap between two otherwise incompatible policies.

We focus the introduction to our approach on the most challenging case (Type C transition), which requires intermediate policies. Note that the need for intermediate policies is specific to the subtask definition. Our goal in this work is to provide solutions to three types of transitions, so the user can systematically build transitions between any arbitrary pair of subtasks. Intermediate policies are needed for Type-C transitions only. As shown in Figure 2, most transitions (Type A and B) do not require them and can be handled by simplified variants of the same method (Figure 3). We ground our exposition on the task of *shooting off the dribble*, a common basketball skill. By introducing an intermediate gathering policy between the pre-trained dribbling and shooting policies, our simulated player achieves a 91.8% shooting success rate from a wide range of challenging dribbling states, including facing away from the hoop and performing complex maneuvers such as spins, pivots, and pull-up jumpers.

The conditions for transition are also crucial to the success of a challenging long-horizon task. We introduce a high-level routing policy, trained to automate subtask transitions, eliminating the need for manually defined state conditions. Similar to controls in video games, the router allows transitions to be triggered by a user or external program, balancing smoothness with responsiveness.

Although prior work has demonstrated that imitation-based RL are effective in learning from human motions, ensuring their robustness and generality remains challenging and requires substantial demonstration data. Unfortunately, structured datasets with corresponding full-body motion, detailed finger motion, and ball trajectories remain scarce. Therefore, a secondary but key contribution of our approach is demonstrating that it is possible to learn

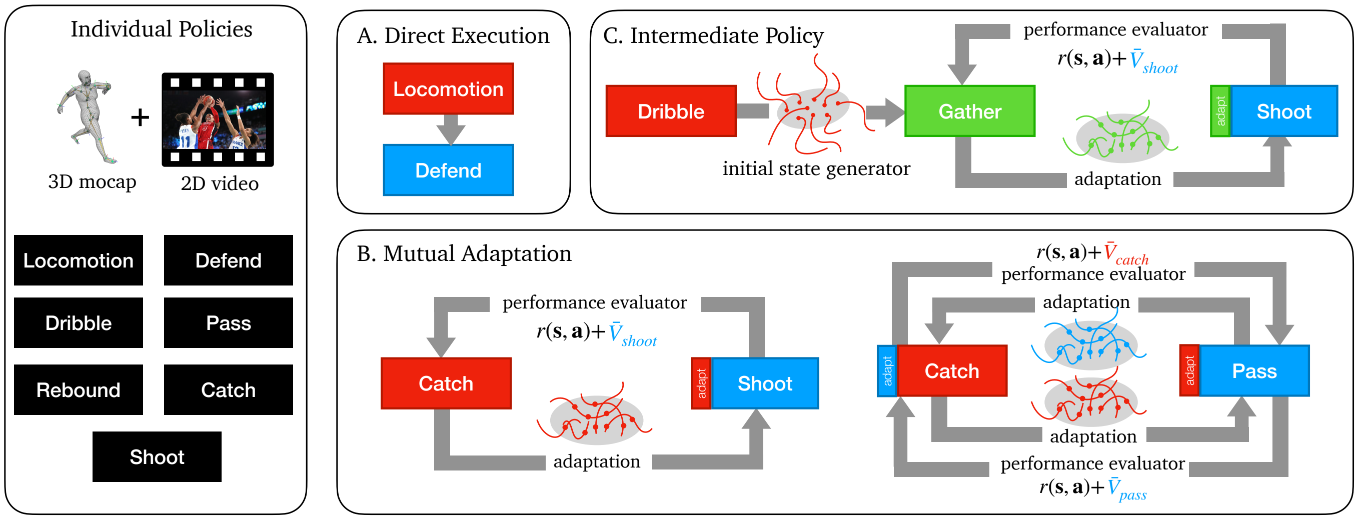


Fig. 3. Three transition types are illustrated by examples. This work focuses on the most challenging case, Type C transition (upper-right), which requires an intermediate policy to facilitate the transition. To train a policy for such an ill-defined subtask (green), our method utilizes the terminal states of the preceding policy (red) to provide an initial state distribution, and the succeeding policy (blue) to provide its state value function $\bar{V}_{\text{shoot}}$ for reward shaping. Simultaneously, the selected states from the rollouts of the intermediate policy are used to adapt the succeeding policy. Type B transition is less challenging and can be done with only adaptation and value-function-based reward shaping, without the need to train an intermediate policy specifically. Type A transition is the least challenging one and can be done by directly executing the succeeding policy from any state of the preceding policy.

complex basketball skills from heterogeneous datasets containing both 2D videos and 3D mocap, without requiring ball trajectories or correspondence between full-body and hand motion data.

4 Learning from Unstructured Motions

Figure 4 shows our system architecture for primitive policy learning from unstructured motions. We use Proximal Policy Optimization (PPO) [Schulman et al. 2017] as the backbone reinforcement learning algorithm and employ an adversarial imitation learning framework [Xu and Karamouzas 2021; Xu et al. 2023a] to train the primitive policies. Specifically, for the *shooting-off-the-dribbling* task, we train policies for two well-defined subtasks: dribbling in an arbitrary target direction with a randomly given speed, and shooting the ball into the hoop. Our objective is to train a character capable of dribbling and shooting on demand in real time. This goal makes it impractical to rely on tracking a fixed set of motion trajectories, as done in prior work [Wang et al. 2024e]. Achieving the desired flexibility requires access to a large-scale, diverse set of reference motions, which necessitates relaxing the assumption of structured motion datasets with one-to-one correspondences between gross body motion, detailed hand motion, and basketball trajectories. To address this challenge, we: a) incorporate multiple motion data sources, including 3D motion capture and 2D video data; b) decouple full-body motions into several groups [Liu and Hodgins 2018; Xu et al. 2023a], reducing the need for combined motions across different body parts; and c) rely on task rewards to eliminate dependency on corresponding basketball trajectories, as collecting or generating them is highly challenging. We refer to the supplementary materials for implementation details and hyperparameters for policy training. Despite the strength of our approach for primitive policy learning, our method for policy transition (Section 6) does not have a special requirement on how primitive policies are trained, and can be combined with other existing, primitive skill learning methods.

4.1 Unstructured Data Sources

Public motion capture datasets, such as LAFAN1 [Harvey et al. 2020], AMASS [Mahmood et al. 2019], the CMU Mocap Dataset [CMU 2003], provide full-body motion data but lack detailed hand motions, including wrist and finger dynamics. On the other hand, numerous datasets focus on detailed hand motions but do not include corresponding full-body motion data [Chao et al. 2021; Fan et al. 2023; Taheri et al. 2020; Wang et al. 2024c]. Video data offer the potential to capture both full-body and hand motions at scale. However, it is limited by challenges such as occlusion, depth ambiguity, inconsistent picture quality, and motion blur, making it less reliable as a primary data source. In this work, we use all the data sources described above, including:

- Internet videos: We use ExAvatar [Moon et al. 2025] and TRAM [Wang et al. 2025] to extract hand and body poses from online videos, respectively.
- Full-body-without-hand mocap data: running motions from LAFAN1 [Harvey et al. 2020] and other motions from the CMU Mocap Dataset [CMU 2003] to train the base skills.
- Hand-only data: We recorded our own finger motions using Rokoko Smartgloves, capturing a subject shooting in place and dribbling within a short range.

Notably, we do not assume access to the corresponding ball trajectories in our dataset. While it is technically feasible to extract ball trajectories from videos or record ball states using motion

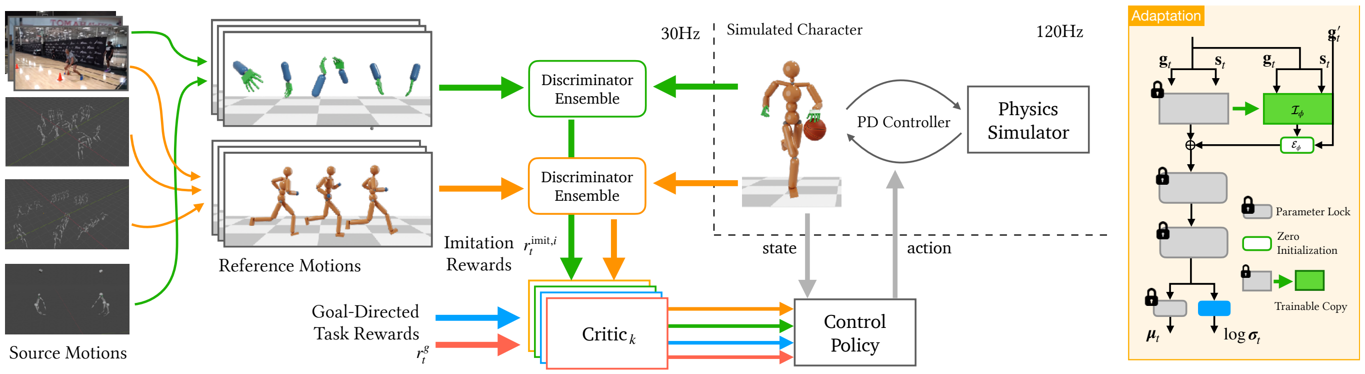


Fig. 4. System architecture for primitive policy learning in our system. We decouple the full-body motions into body and hand split, and use reinforcement learning to imitate unstructured motions collected from disparate sources without needing ball trajectory reference or requiring the correspondence of the motions. From top to down on the left side, we show the screenshots of motions from online videos, body-only mocap data of basketball playing and normal locomotion, and our own captured hand motions. On the right side, we show the network structure for policy adaptation [Xu et al. 2023b]. This structure allows us to introduce an optional additional goal input $\mathbf{g}_t'$ during adaptation for policy transition training, and is suitable for our training strategy for pivoting foot control when transitioning to the shooting policy from dribbling motions (see the supplementary materials for the details).

capture systems, acquiring high-quality and large-scale trajectories remains labor-intensive. Moreover, avoiding dependency on basketball-specific trajectories allows us to utilize non-basketball motion data, such as the running motion from LAFAN1 dataset.

4.2 Learning to Dribble

To fully take advantage of the unstructured motion datasets for training a dribbling policy, we group full-body poses into three categories: lower body, upper body, and hands. Unlike typical grouping schemes [Bae et al. 2023; Liu and Hodgins 2018], we treat the two hands (including wrist rotations) as a separate group from the arms, with the elbows serving as the root links. This separation facilitates the use of hand-only motions. Putting two hands into one group improves coordination between the hands and helps prevent unnatural poses, such as dribbling with both hands simultaneously.

We train the dribbling policy using reinforcement learning in a GAN-like architecture, combined with a multi-objective learning framework [Xu et al. 2023a] to balance imitation and task-specific goals. The policy is guided by two imitation objectives, one for hands and one for the reset body parts, with rewards provided by discriminators that process partially observable motions, as shown in Figure 4. Additionally, two task-specific rewards are employed: one for velocity-controlled navigation and another for dribbling. We refer to the supplementary materials for more details. During training, the policy autonomously explores physical interaction with the ball while being guided by the two task rewards and partially observable reference motions.

To adhere to the basketball rules and ensure valid interactions between the simulated character and the ball, we implement two types of violation detection as part of our goal-directed reward functions that consider: (1) invalid contact between the ball and other body parts besides the hands; and (2) traveling when the ball is held. The input to the dribbling policy includes the current pose of the character and of the ball, a target velocity $\mathbf{v}_{\text{target}}$, and a variable indicating the dribbling state of the ball. The target velocity is generated randomly during training and given by the user via the joystick during interactive control.

4.3 Learning to Shoot

For training the shooting policy, we do not consider any body part grouping but directly perform imitation learning using three full-body demonstrations of jump shooting. We use one full-body imitation objective and a task-specific reward measuring the shot accuracy and ball-holding performance before the shot is taken. We refer to the supplementary materials for the details of state space, action space, and reward definition.

5 Learning Intermediate Subtasks

In basketball terms, the transition between dribbling and shooting is defined by another subtask called "gathering". This intermediate but critical subtask not only requires the character to rapidly stop and catch the ball, but also to adjust ball-hold poses and body orientations for jump shooting while maintaining balance. However, gathering does not have clearly defined initial and terminal states, preventing simply training an imitation policy or chaining it between the dribbling and shooting policies.

To train such an intermediate subtask, our method utilizes the terminal states of rollouts generated by the preceding policy (dribbling) to provide an initial state distribution. Without defining a fixed set of terminal states of dribbling, the gathering policy is expected to take over the character from any dribbling state. Thereby, random states drawn from the dribbling rollouts will be taken as the initial state for the gathering policy. Meanwhile, we utilize the state value function of the succeeding policy (shooting) to shape the reward function, so that the character learns to reach a state from which the shooting policy is likely to succeed (cf. Type C in Figure 3). The reward for the gathering policy is defined as:

$$r_{\text{gather}} = \begin{cases} -1 & \text{if any violation is detected,} \\ r_{\text{pose}} + 0.25\textsc{Clip}\left(\bar{V}_{\text{shoot}}(\mathbf{s}_t, \mathbf{g}_t), -v, v\right) & \text{otherwise.} \end{cases} \quad (1)$$

$r_{\text{pose}} \in [0, 1]$ is a reward term evaluating the ball holding performance, involving finger and palm poses related to the ball and body orientation related to the hoop (see the supplementary materials for details). We define $\bar{V}_{\text{shoot}}$ as the accumulated *task* reward from state $\mathbf{s}_t$ following the shooting policy, bounded by $[-v, v]$. We employ PopArt [Van Hasselt et al. 2016] to perform value normalization on the critic (value network) output in PPO to obtain $\bar{V}_{\text{shoot}}$. We set $v = 1$ in all of our experiments. We refer to the supplementary materials for details of the reward function definition. While the $\bar{V}_{\text{shoot}}$ term can help guide the policy to generate poses preferred by the shooting policy, the heuristic reward term r_{pose} for gathering is still necessary to steer the character to a near-shooting pose, since the value function will generally produce low values when the character is far from shooting poses, leading to ineffective guidance.

To improve the generalizability of the shooting policy and make it better cooperate with the gathering policy, we further adapt the shooting policy, in tandem with the gathering policy training, by taking the states produced by the gathering policy as the initial state for the shooting policy, using the approach of latent space manipulation from AdaptNet [Xu et al. 2023b].

Instead of adapting the shooting policy *after* the gathering policy is trained, a key algorithm design choice is to adapt the shooting policy simultaneously *during* training of the gathering policy. Specifically, we filter out the "good" states encountered during the training of the gathering policy and use them as the initial states for adapting the shooting policy. As the policy is adapted, its corresponding state value function is also updated and being used by the reward function of gathering policy. A good state has a score greater than $-v$ when evaluated by the evolving $\bar{V}_{\text{shoot}}$. Since initially the poses generated by the gathering policy are distinctively different from the high-value states for the pre-trained shooting policy, we bootstrap the learning by randomly permitting 25% of the states with value estimations less than $-v$, as long as the ball is held in hand and the character is approximately facing the hoop. These states are used as additional initial states to adapt the shooting policy.

The gathering policy is trained using the same goal state as we obtain the pretrained shooting policy (Section 4.3) but with an additional variable to indicate the pivoting foot to prevent traveling. We ignore the traveling problem during the pertraining of the shooting policy as it would not happen when the policy simply imitates the shooting motions in the reference, where the subject directly jumps up for shooting. With the introduction of gathering, we extend the shooting policy during adaptation and also include the pivoting foot indicator in the goal state of the adapted shooting policy.

6 Composing Policies for Long-Horizon Tasks

One possible approach to transitioning between policies is to initiate gathering when the user commands the character to take a shot, and transition to the shooting policy when the reward of the gathering policy (Eq. 1) is higher than the $-v$ threshold. However, using such a heuristic often results in the character stalling in a ball-holding pose without transitioning to the shooting policy or failing to catch the ball after the transition. To address this issue, we introduce a high-level routing policy that assembles the subtask policies to perform a long-horizon task (Figure 5).

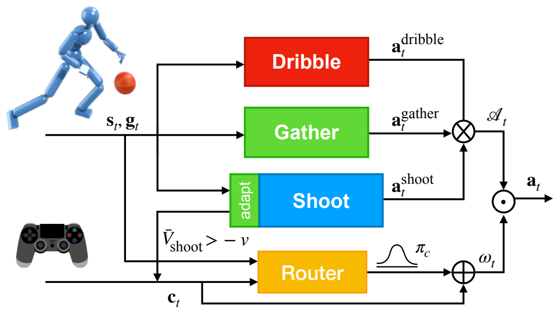

Dribble
Gather
Shoot
adapt
Router
$\mathbf{a}_t^{\text{dribble}}$
$\mathbf{a}_t^{\text{gather}}$
$\mathbf{a}_t^{\text{shoot}}$
$\mathscr{A}_t$
$\mathbf{a}_t$
$\mathbf{s}_t, \mathbf{g}_t$
$\bar{V}_{\text{shoot}} > -v$
π_c
ω_t
$\mathbf{c}_t$

Fig. 5. The high-level routing policy assembles the subtask policies to perform shoot-off-the-dribble task. It takes as input the user command, the current state, and the goal vector, and outputs the weights for linear combination of the actions from the subtask policies. ⊕ denotes element-wise add, ⊗ denotes concatenation, and ⊙ denotes dot product.

The primitive policies in our studied case synthesize the unstructured motions for subtask execution instead of tracking a fixed set of or a given full-body reference. This makes motion imitation, during high-level policy training, inapplicable. To avoid the high-level policy averaging too much on the outputs of the primitive policies and generating unnatural motions, we model the high-level policy as a router function. Unlike the conventional policy routing technique [Bacon et al. 2017; Tessler et al. 2017], our routing policy performs soft merging instead of activating only one policy at a time step. We define a reference command $\mathbf{c}_t$, as a one-hot vector to heuristically indicate whether the character should dribble, gather, or shoot the ball. The routing policy takes $\mathbf{c}_t$ as input and outputs an offset from $\mathbf{c}_t$ to produce the weights for linear combination of the actions from the primitive policies:

$$\boldsymbol{\omega}_t = \mathbf{c}_t + \pi_{\mathrm{c}}(\mathbf{s}_t, \mathbf{g}_t, \mathbf{c}_t), \quad \mathbf{a}_t = \boldsymbol{\omega}_t \cdot \mathscr{A}_t. \tag{2}$$

Here π_{c} is the routing policy and $\mathscr{A}_t = [\mathbf{a}_t^{\text{dribble}}, \mathbf{a}_t^{\text{gather}}, \mathbf{a}_t^{\text{shoot}}]$ is the collection of the deterministic output from the three subtask policies, i.e. $\mathbf{a}_t^{\text{dribble}} = \mathbb{E}[\pi_{\text{dribble}}(\cdot|\mathbf{s}_t, \mathbf{g}_t)]$, $\mathbf{a}_t^{\text{gather}} = \mathbb{E}[\pi_{\text{gather}}(\cdot|\mathbf{s}_t, \mathbf{g}_t]$, $\mathbf{a}_t^{\text{shoot}} = \mathbb{E}[\pi^+_{\text{shoot}}(\cdot|\mathbf{s}_t, \mathbf{g}_t)]$ and π^+_{shoot} indicates the shooting policy after adaptation. Note that the definition of $\mathbf{g}_t$ with respect to each subtask is different. We refer to the supplements for details.

The reference command for dribbling is $\mathbf{c}_t = [1, 0, 0]$ and becomes $[0, 1, 0]$ after receiving an external command to shoot. When $\bar{V}_{\text{shoot}}(\mathbf{s}_t, \mathbf{g}_t) > -v$, we set $\mathbf{c}_t = [0, 0, 1]$, which implies that the shooting policy may be able to take over the character. At the beginning of training, π_{c} outputs small values and the exploration will start based on the given $\mathbf{c}_t$ as guidance. As training goes on, the final weight $\boldsymbol{\omega}_t$ may diverge from $\mathbf{c}_t$ in order to generate more stable transitions between different policies.

To encourage π_{c} to generate one-hot-like weights $\boldsymbol{\omega}_t$ through which only one policy dominates at each time step, we define the training reward for π_{c} as follows:

$$r = \begin{cases} r_{\text{gather}} & \text{if the gathering policy dominates the control,} \\ 0.5 I_{\text{switch}} + r_{\text{shoot}} & \text{if the shooting policy dominates,} \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where we consider a policy dominates the control if its associated weight in $\boldsymbol{\omega}_t$ is larger than the sum of the other two. $I_{\text{switch}} = 1$ only for the first time when the dominant policy switches from *gather* to *shoot* and $I_{\text{switch}} = 0$ otherwise. The detailed definitions of r_{gather} and r_{shoot} are provided in the supplementary materials. The high-level policy after training can achieve almost 100% success rate for ball gathering from dribbling states and an overall shot percentage of 91.8% (see Section 8).

To reduce the inference time consumption, after the training of the high-level composer policy, we perform an additional distillation process to compress the hierarchical policy into a single neural network. This process is done in a simple supervised learning manner with samples generated online through the physical simulator. The distilled policy during our experiments can achieve the same performance as the hierarchical policy with only trivial errors.

7 Other Skill Transitions

The same method used for *shoot-off-the-dribble* is directly applied to *pass-off-the-dribble*. Both of them are defined as Type C transitions as shown in Figure 3. Our system also supports a range of other skills, including rebounding, catching, defending, general locomotion, and various transitions, as summarized in Figure 2. Type A transitions are achieved through direct execution of the succeeding policy. Type B transitions serve as a simplified version of Type C, requiring reward shaping of the preceding policy using the value function of the succeeding one, along with adaptation of the policy using rollouts generated by the preceding one (cf. Figure 3).

A variation of Type B transitions is employed when training catching and passing transitions (Figure 3, bottom-right). After individually pre-training both policies, we fine-tune them jointly through co-adaptation and co-reward shaping using two interacting agents. While adapted to diverse initial poses produced by the terminal states of the passing and catching policies, this enables the passing agent to learn to make catchable throws, and let the catching agent learn to receive the ball in a state ready for an immediate pass/shot.

8 Experiments

We conduct experiments to evaluate our skill composition framework for physics-based character control. While we provide qualitative results for all skill transition types in Section 8.3 (see also companion video), our quantitative analysis in Section 8.2 focuses on the most challenging Type C transition from dribbling to shooting that requires the use of an intermediate gathering policy, as shown in Figure 3. Our experiments utilize IsaacGym [Makoviychuk et al. 2021] as the underlying physics engine. All policies are trained using PPO [Schulman et al. 2017]. Other implementation details, including policy training procedures and hyperparameters, are provided in the supplementary materials.

8.1 Data Preprocessing

As detailed in Section 4, we use ExAvatar[Moon et al. 2025] and TRAM [Wang et al. 2025] to extract 3D hand and body poses from videos and perform motion synthesis via imitation learning for primitive policy training. The video data is collected exclusively from publicly available sources. Additionally, we enrich the dataset by combining the processed motions with publicly available mocap data and a small set of hand motions that we captured ourselves.

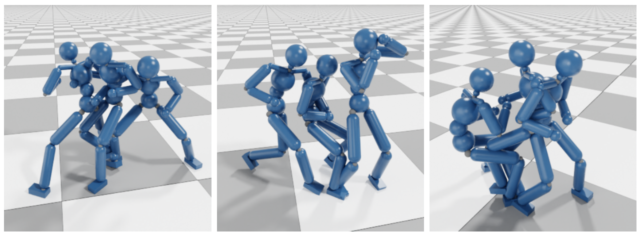


Fig. 6. Examples of the gathering reference motions extracted from videos without hand. From left to right, the subject is respectively pivoting to spin clockwise with the right foot on the ground, catching the ball directly during dribbling forward for shooting, and performing a fast in-place spinning counter-clockwise with the right foot pivoting on the right foot.

Table 1. Reference motions for primitive policy training. All videos are collected online. The **Dribble** mocap (body) data is collected from CMU Mocap dataset. **Locomotion** data are running motions from LAFAN1. All mocap (hand) motions are collected by ourselves. Additionally, we take 3 demonstrations of jump-shooting motions (full body with hands) to train the shooting policy. We mixedly use gathering and catching motions for catching and rebounding. All motions do not include ball trajectories.

Source	Length	Source	Length
Dribble	*(body/hand)*	**Defensive Stance**	*(body/hand)*
video	48.8s/36.1s	video	9.3s/1.4s
mocap	58.2s/30.5s	mocap	25.1s/25.1s
Gather	*(body/hand)*	**Locomotion**	*(body)*
video	30.3s/7.7s	mocap	141.4s
Catch	*(body/hand)*	**Pass**	*(full-body)*
video	10.1s/10.1s	video	4.5s

Table 1 summarizes the data sources and the motion lengths in our reference motion dataset. Most of the **Dribble** motions extracted from videos consist of in-place movements due to the limitation of the pose estimation models when facing fast movement subjects. We incorporate normal running motions from LAFAN1 [Harvey et al. 2020] to enable the policy to learn locomotion with fast movements and directional changes during dribbling. **Gather** motions are extracted from 13 video demonstrations (see examples in Figure 6). Due to occlusion and rapid hand movements, we obtained only a limited set of ball-holding hand motions during the gathering phase. The **Defensive Stance** motions include stance poses while the subject stretches the arms to perform blocking or screening. For shooting, we specially use three full-body demonstrations of jumping shooting, including hand poses. Our dataset does not include ball trajectories. Instead, the control policy explores valid interactions with the ball on its own under the guidance of the task rewards.

8.2 Quantitative Evaluation

To evaluate the quality of Type C transitions generated by our approach, we analyze the performance of the shoot-off-the-dribble task by focusing on two primary metrics:

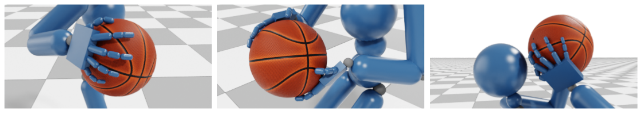


Fig. 7. Learned ball-hand interactions by policies trained with our approach. From left to right, the snapshots are captured during the character dribbling, gathering and shooting the ball, respectively.

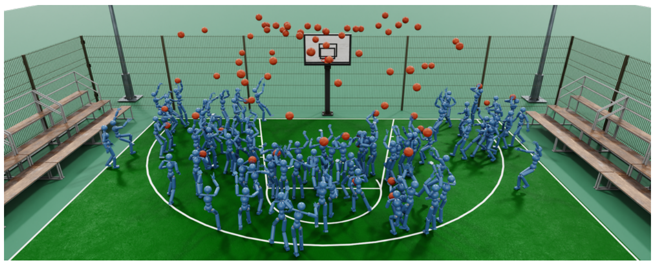


Fig. 8. The testing scenario to count the shot percentage.

- *Ball Catching Rate:* This metric checks whether the ball is successfully gathered and held by the character after a shooting command is issued and before the ball is shot.
- *Shot Percentage:* This measures the ratio of successful field goals to attempted field goals. Attempts that fail due to unsuccessful ball catching or a lack of response to the shooting command are also included in this metric.

We compare our method against three baselines which offer alternative approaches for policy composition. These baselines represent *zero-shot* and *sequential* approaches that have been explored in different application domains for skill chaining.

- ***DirectExecution*** *(Dribble⟶Pretrained Shoot):* This baseline uses the pretrained shooting policy (π_{shoot}) without any adaptation or intermediate gathering policy. We use grid search on the state value of the shooting policy to find out the optimal transition, which is similar to the implementation in prior work [Xu and Karamouzas 2021].
- ***NoAdapt*** *(Dribble⟶Gather⟶Pretrained Shoot):* This approach incorporates a gathering policy but no adaptation is applied to π_{shoot}. The function of $\bar{V}_{\text{shoot}}$, thereby, is fixed during gathering policy training. Like *DirectExecution*, this is zero-shot with the state value used for policy switching.
- ***SequentialChaining*** *(Dribble⟶Shoot) :* The sequential approach leverages terminal states from the preceding policy to train the succeeding policy and improve transition success [Clegg et al. 2018; Liu and Hodgins 2017; Wang et al. 2024e]. A shooting policy is trained from random initial states provided by the dribbling policy.

Additional comparisons to other policy-switch strategies and detailed ablation studies are provided in Section 9.

8.2.1 Ball Catching Rate. Figure 9 shows the ball catching rate for different methods as a function of the character's dribbling speed at the time a shooting command is issued. Our method shows robust performance across all speeds, maintaining a high ball catching rate, while the baselines exhibit significant limitations.

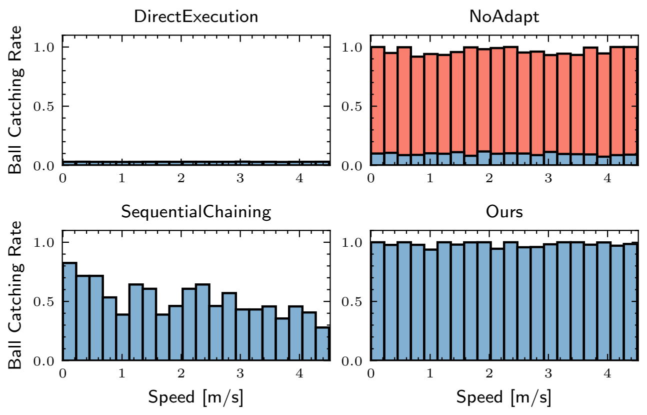


Fig. 9. Ball catching rates as a function of dribbling speed. The **DirectExecution** baseline fails almost entirely due to the incompatibility between policies. The **NoAdapt** baseline shows high success rates for the gathering policy (red bars), but suffers from poor generalization of the shooting policy after the shooting policy takes over the character (blue bars). The **SequentialChaining** baseline performs better overall but underperforms compared to the dedicated gathering policy in the **NoAdapt** method. Our method achieves consistently high ball catching rates across speeds.

The **Direct Execution** baseline fails almost entirely, with a 0.7% catching rate across all speeds. This result highlights the incompatibility between the dynamic dribbling state and the unadapted shooting policy, consistent with prior findings [Xu and Karamouzas 2021] that direct policy switching requires the preceding policy to transition the system into a compatible state for the succeeding policy. For the **NoAdapt** baseline, the red bars in Figure 9 indicate that the gathering policy performs well at catching the ball, achieving a high average catching rate across all speeds. However, the pretrained shooting policy struggles to generalize, resulting in significant performance drop. Even when adapting the shooting policy during the training of the gathering policy (as shown in our supplementary materials), the average ball catching rate remains low unless the state value is explicitly incorporated as a reward term for the gathering policy (cf. Eq. 1). This result underscores the difficulty of achieving effective cooperation between the gathering and shooting policies through random exploration alone during training. The **SequentialChaining** baseline, which trains a single policy for both gathering and shooting, achieves a higher average ball catching rate than the **DirectExecution** baseline but underperforms compared to the **NoAdapt** that utilizes a separate gathering policy. This degradation suggests that the lack of explicit phase division between gathering and shooting hinders policy effectiveness.

In contrast, our method achieves a 98.3% ball catching rate across all dribbling speeds, as shown in the bottom-right plot of Figure 9. By dividing the task into distinct phases and leveraging state value as a reward signal, our framework ensures smooth cooperation between policies, enabling reliable ball gathering and shooting even under highly dynamic conditions. These results underscore the value of structured phase division and reward shaping in multi-phase tasks.

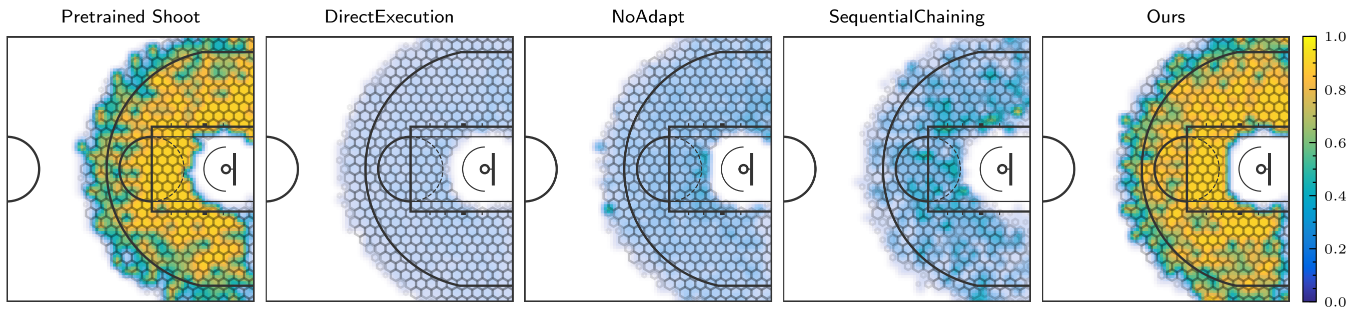


Fig. 10. Heatmap of shot percentages for the baselines and our method across different positions and approaching directions on the court. The first subplot shows the performance of the vanilla pretrained shooting policy, where the character and ball states are initialized using reference motions rather than random dribbling states, achieving an overall shot percentage of 93.0%. For baselines, directly applying the pretrained shooting policy or using combined gathering and shooting strategies leads to poor performance (1.3%, 6.1%, and 12.7% overall shot percentages, respectively), as they fail to handle the dynamic and diverse states arising from dribbling. Our method achieves a shot accuracy of 91.8% nearly matching the vanilla pretrained policy, demonstrating its ability to adjust character poses dynamically during the gathering phase and perform accurate shots even in challenging scenarios.

Table 2. Shot percentage of our approach when the character dribbles at different approaching directions toward the hoop. "Orth." stands for the orthogonal direction. Each of the reported numbers is obtained by considering an angle range of $\pi/2$. The overall shot percentage is 91.8%. We refer to the supplementary material for visualized results.

	Facing	Opposite	Left Orth.	Right Orth.
Shot Percentage	95.4%	90.4%	92.7%	92.4%

8.2.2 Shot Percentage. We evaluate the shot percentage with the character under varying positions and approaching directions toward the hoop. Similar to the training setup, we define a valid shooting area as a ring between 2.5m and 7.5m from the hoop, as shown in Figure 10. This area is divided into a grid, where each grid cell has a radius of 0.5m. In each cell, we conduct 400 trials with the character dribbling at different speeds and approaching directions. While our approach has an overall shot percentage of 91.8%, **SequentialChaining** achieves a slightly higher shot percentage (12.7%) than the other two baselines, but still performs much poorly compared to our method. According to Figure 9, our method has a high success rate of ball catching. Occasional failures may occur due to excessive movement speed or an extremely unstable state (e.g., fast spinning). For shooting off the dribbling, most missed shots originate from positions that are too far from the hoop, as shown in the rightmost subplot in Figure 10, instead of the policy transition.

Additionally, Figure 10 includes the vanilla performance of the pretrained shooting policy, where the character and ball states are initialized using shooting reference motions rather than random dribbling states. Despite being trained with only three demonstrations of shooting motions, the vanilla pretrained shooting policy achieves a high overall shot percentage in a wide area around the hoop. This success highlights its effectiveness when starting from well-aligned and familiar states. However, when applied to unseen character and ball states resulting from dribbling or gathering, the policy's performance deteriorates significantly.

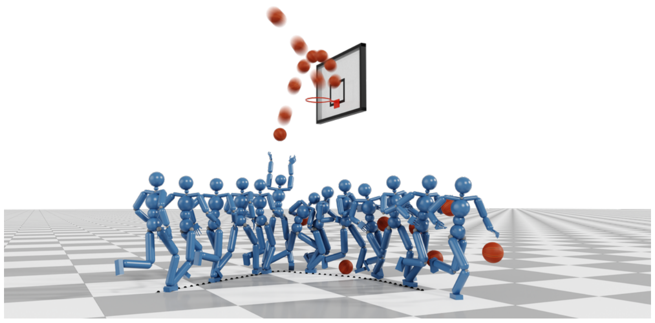


Fig. 11. Transitions from locomotion to rebounding and to dribbling. The former transition is obtained via the common stance poses without additional learning (Type A transition). The latter one is obtained by adapting the pretrained dribbling policy using terminal (ball-holding) poses from a catching policy (Type B transition).

Table 2 further analyzes the shooting performance of our approach based on the character's approaching direction, demonstrating its adaptability across a wide range of scenarios. As can be seen, the shot percentage when the character approaches the hoop in a facing direction is better than the vanilla pretrained policy. Overall, our solution enables the policy to handle diverse and dynamic states effectively, allowing the character to adjust its pose during gathering. This includes spinning, pivoting, or turning to align with the hoop, even when the character initially faces the opposite direction to the hoop.

8.3 Qualitative Analysis

Despite the absence of ball trajectory data and the reliance on normal running motions to learn locomotion during dribbling, the dribbling policy successfully synthesizes unstructured motions from multiple sources. It enables the character to perform complex tasks such as sharp turns and abrupt stops while maintaining effective ball control for arbitrary target velocities in the range of 0 to 5 m/s. We refer to the supplementary materials for the demonstrations along

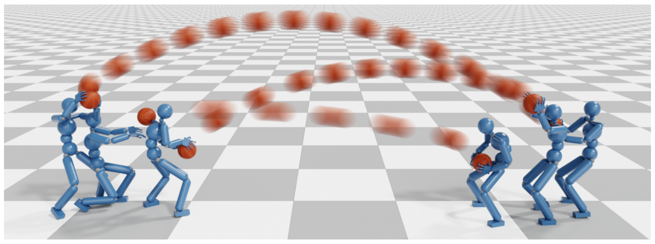


Fig. 12. Passing and catching transition between different agents. The two policies are adapted using the other's state value function simultaneously.

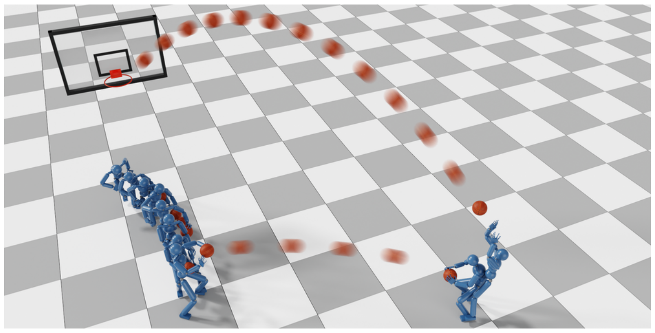


Fig. 13. Cooperative behaviors between two characters. Both of the passing-off-the-dribbling and shooting-off-the-catching behaviors are achieved by adapting and composing the corresponding primitive policies using our presented pipeline.

with those of the other primitive skills trained using adversarial imitation learning. In all these cases, except for the shooting and passing skills, we decouple full-body motions, enabling training on unstructured, partially observable motions collected from disparate sources without needing to track any reference ball trajectories.

Figure 14 illustrates the transitions from various dribbling states to shooting. The character is able to adjust dynamically based on context: catching the ball swiftly when possible, maintaining a dribble-like state during sharp turns before gathering the ball, or pivoting by spinning around one foot to align with the hoop. As shown in Figure 6, the ball-gathering motions are notably different from those used for dribbling or shooting. Mixing gathering reference motions with shooting motions during policy training leads to undesirable behaviors, such as directly throwing or punching the ball toward the hoop instead of shooting (see the supplementary video).

Unlike shooting off the dribble, the transitions shown in Figure 11 are obtained without training an intermediate policy. Here, the rebounding policy is executed directly from the locomotion policy. The same applies to the transition from rebounding to dribbling, though the succeeding dribbling policy is adapted with a ball-catching policy. Similar adaptation is applied to improve cooperative behaviors between two interacting characters by mutual adaptation, such as passing and catching the ball, and passing-off-the-dribbling and catch-and-shoot as shown in Figures 12 and 13. Our system also supports competitive interactions between two characters by training a defending policy that enables strategic movements and defensive

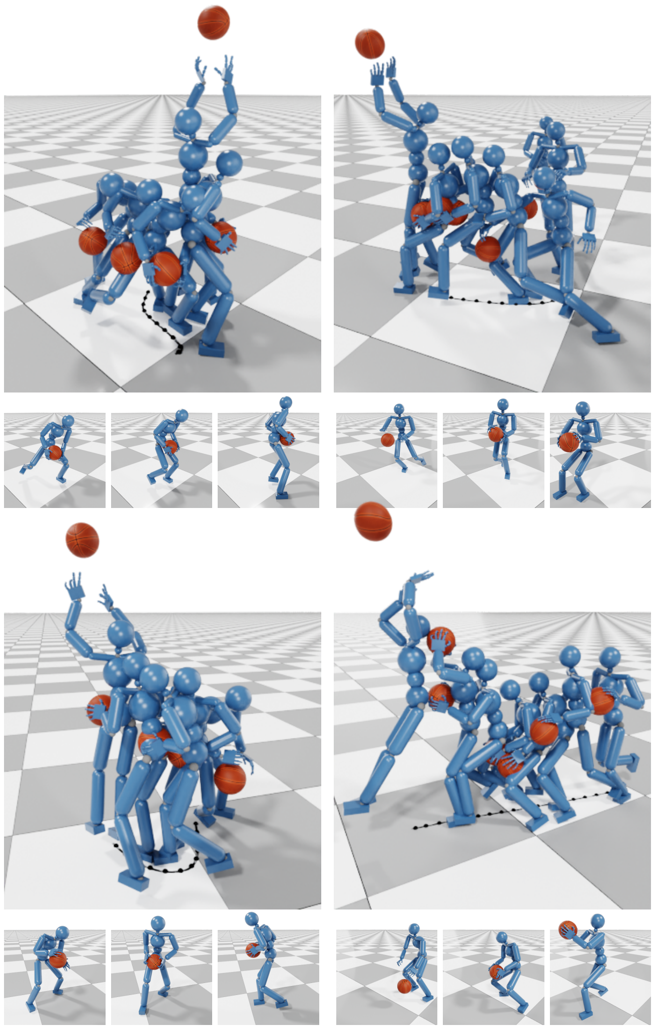


Fig. 14. Demonstrations of our approach that controls the character to shoot off the dribble at arbitrary dribbling states. The snapshots under each subfigure show the keyframes where the character gathers the ball and adjusts its pose for shooting.

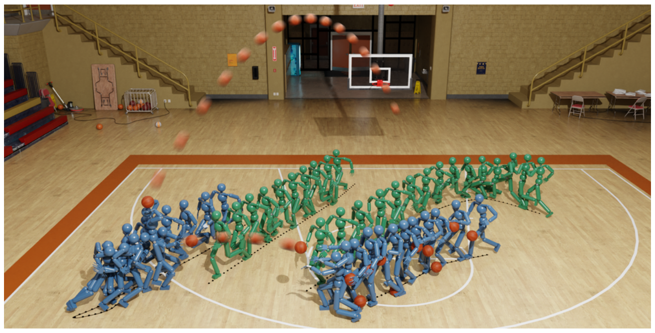


Fig. 15. Four characters are controlled by human players through our control policies to perform a 2-on-2 competition in real-time.

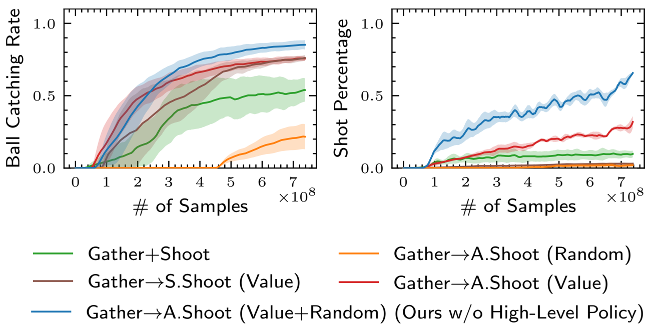


Fig. 16. Learning Performance with different intermediate policy training strategies. "S." stands for "Scratch" and means a shooting policy trained from scratch with the gathering policy. "A." stands for "Adapted" and means a pretrained shooting policy adapted with the gathering policy. "Value" and "Random" indicate the way to transfer the character and ball states produced by the gathering policy to train or adapt the shooting policy. "Value" means to transfer based on the state value evaluation from the shooting policy and "Random" means to randomly transfer states from gathering to the shooting policy. In the case of "Random", the gathering policy serves only as a sample generator for the shooting policy training or adaptation, and the state-value term in Eq. 1 thus will be ignored. The shaded ranges show the performance over three training trials.

actions such as screening and blocking by raising hands up. Figures 1 and 15 showcase interactions in which the offender players (blue) shoot while the defenders (green) block.

9 Ablation Studies

We focus our ablation studies on the evaluation of (1) the training strategy to obtain a good intermediate policy that bridges the gap between a preceding policy (dribbling) and a succeeding policy (shooting), and (2) the effectiveness of the high-level routing policy for primitive policy composition.

9.1 Intermediate Policy Training

In Figure 16, we compare the learning performance of different strategies for intermediate gathering policy training and shooting policy fine-tuning. Gather+Shoot is the **SequentialChaining** baseline, as we discussed in Section 8, which treats ball gathering and shooting as a whole task in one phase. Only a single policy is trained for that baseline, using a mixture of the gathering and shooting reference motions. All the gathering policies shown in the figure are trained to utilize the same pretrained dribbling policy as an initial state generator, and all the adapted shooting policies (A.Shoot) perform adaptation based on the same pretrained shooting policy. The state-value based term in the reward function r_{gather} (cf. Eq. 1) will be ignored in the baselines without the "Value" label. Similar to results shown in Section 8, the ball-catching rate is evaluated not only for the gathering policy, but also for the shooting policy to see if the transfer from gathering to shooting is stable. To draw a fair comparison, here we show the performance of our system without introducing the high-level policy for policy composition.

As shown in the figure, Gather→A.Shoot (Random) performs the worst among the baselines, with only a small improvement in ball catching rate at the end of training. We also test an additional baseline (not shown in the figure) using a random state transferring strategy for the shooting policy trained from scratch, i.e. Gather→S.Shoot (Random). It performs even worse without any performance improvement throughout the training. In contrast, the "Value" baselines show a much more stable performance of ball catching rate. It means that the introduction of the state-value reward term can effectively help the gathering policy learn how to control the character reaching a state manageable by the shooting policy. Comparing the performance of Gather→S.Shoot and that of Gather→A.Shoot, while the ball-catching rate is similar when state-value based transfer is adopted, policy fine-tuning based on a pretrained shooting policy (A.Shoot cases) shows better performance on shot percentage.

The state-value evaluation from the shooting policy can effectively improve the performance of the gathering policy. However, the state-value estimator (the value network from the shooting policy under training for S.Shoot baselines or adaptation for A.Shoot baselines) is trained in tandem with the gathering policy, and is not always reliable, as it may be stuck at local optimal and cannot effectively evaluate unseen states or potentially good states (the states that are acceptable to make a field goal but the shooting policy has to be further trained or adapted to work with the given states). Therefore, to increase the robustness of the whole system, our method takes a random sampling strategy complementing the state-value based transfer strategy for shooting policy adaptation, as described in Section 5.

Gather+Shoot shows worse performance compared to the baselines of Gather→A.Shoot (Value), which demonstrates the necessity to explicitly set multiple phases for transitions between drastically different subtasks. Using the gathering and shooting reference motions simultaneously could lead to unnatural behaviors, where the character, for example, throws out the ball while back facing the hoop or directly punches the ball out toward the hoop. We refer to the supplementary video for the animated results.

9.2 High-Level Routing Policy

We compare the performance of our system with and without the high-level policy in Figure 17 and 18. Similar to other baseline comparisons, when no high-level policy is introduced, we perform policy switches between dribbling, gathering, and shooting in a heuristic way: the gathering policy will be called right after receiving a shooting command, and the shooting policy will take over the character if the state value evaluation based on the current state of the character and ball is higher than a threshold value. The optimal threshold value is found by a grid search in the range $[-1, 0]$ with an interval of 0.1. As shown in the figures, although both methods rely on the same primitive policies, the high-level policy achieves better performance in terms of both the ball catching rate and shot percentage, while also automating the transfer between those primitive policies. Visually, without the high-level policy, heuristic rule-based policy switch would cause undesired behaviors like a delayed response from gathering to shooting, sticking to a ball holding pose without performing shooting anymore, or even the character falling down. We refer to the supplementary video for the animated results.

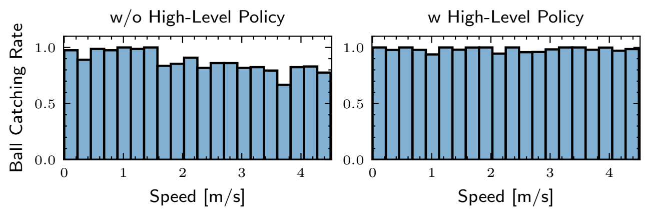


Fig. 17. Ball catching rate of our system with and without the high-level policy while the character dribbles at different velocities. The overall ball catching rate is 86.0% when no high-level policy is introduced, and 98.3% with the high-level policy.

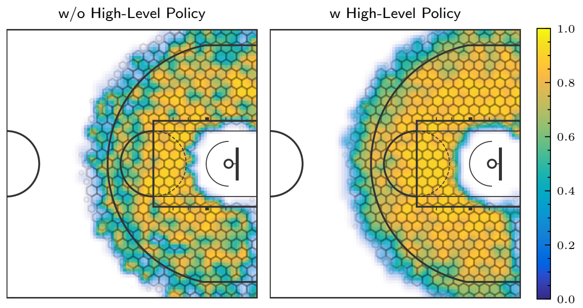


Fig. 18. Heatmap of the shot percentage of our system with and without the high-level policy. The overall shot percentage is 67.4% when no high-level policy is introduced, and 91.8% with the high-level policy.

The high-level policy acts in a mixture-of-the-expert way. To preserve the naturalness of the generated motions and prevent excessive averaging across the output of primitive policies, the high-level policy trained to encourage outputting actions dominated by one primitive policy (cf. Eq. 3). This is a "soft routing" strategy, as it still allows for the high-level policy to slightly merge behaviors from multiple primitive policies. By using the soft routing strategy, we can easily incorporate the reference command to the output of the high-level policy. Training of the high-level policy, thereby, starts with exploration around the reference command generated heuristically in the same way as the baseline shown in Figures 17 and 18. This improves the training efficiency of the high-level policy largely. In Figure 19, we compare the training performance of our soft routing strategy with the hard routing strategy. In the hard routing case, the output of the high-level policy is modeled as a discrete one-hot vector through a softmax operation. Without the guidance from the reference command, the high-level policy in "hard routing" cases has to rely on itself to explore how to compose multiple primitive policies. As shown in the figure, the performance of the hard router is much lower than our soft router, and even worse than the baseline using only heuristic rules (the reference command) without the high-level policy. This highlights that, despite having pretrained primitive policies, learning to effectively compose them remains a non-trivial challenge. Our soft-routing approach provides a more effective way to compose policies in a flexible way.

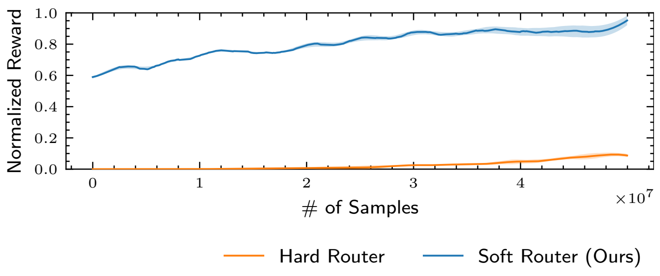


Fig. 19. Learning performance when modeling the high-level policy as a soft router (ours) and that modeling the high-level policy as a hard router.

10 Conclusion

Our results demonstrate the efficacy of a policy integration framework in tackling multi-phase, long-horizon tasks. By treating preceding policies as initial state generators and succeeding policies as evaluators via state value functions, we bridge the gap between drastically different subtasks, enabling seamless transitions in scenarios where intermediate states defy standard definition. This framework is flexible and works in less challenging cases without the need for a newly learned intermediate policy, or in scenarios involving sequential cooperation between multiple agents. The success of our approach in synthesizing diverse motion behaviors—using unstructured data sources and limited reference motions—highlights its generality, with implications beyond basketball. Our method can adapt to characters with different morphologies and skill stats by using styled reference motions and stats constraints during primitive policy training. Once a characteristic-conditioned policy is trained, the transitioning procedure remains the same. The whole policy transition system is agnostic to the task of each primitive policy and can be easily extended by introducing more primitive policies.

Despite the strengths of our approach, this work has certain limitations. The motion quality falls short of the level demonstrated by skilled human players, primarily due to limited reference data. For instance, although ball-hand interactions appear plausible, the character often adopts a high dribbling posture, with hands near chest level—resembling arm positions during regular running. While ball control is reliable and rarely fails, it lacks the fluidity typical of human dribbling. Likewise, because we use only a single clip of a low jumping motion, the character exhibits minimal vertical lift when attempting to rebound the ball.

A potential future work is incorporating a biomechanically accurate hand model, which could encourage more natural physical interactions with the basketball. Another promising direction for future research involves using large language models to plan high-level tactical plans for multiple simulated players. This would allow the training of multiple autonomous agents in cooperative and adversarial settings.

Acknowledgments

This work was supported in part by the Wu-Tsai Human Performance Alliances, Stanford Institute for Human-Centered Artificial Intelligence, and Roblox. We thank Joe Gibbs Human Performance Institute and Rokoko for providing mocap data for testing.

References

Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 31.

Jinseok Bae, Jungdam Won, Donggeun Lim, Cheol-Hui Min, and Young Min Kim. 2023. Pmp: Learning to physically interact with environments using part-wise motion priors. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–10.

Yu-Wei Chao, Wei Yang, Yu Xiang, Pavlo Molchanov, Ankur Handa, Jonathan Tremblay, Yashraj S Narang, Karl Van Wyk, Umar Iqbal, Stan Birchfield, et al. 2021. DexYCB: A benchmark for capturing hand grasping of objects.

Sirui Chen, Albert Wu, and C Karen Liu. 2023b. Synthesizing Dexterous Nonprehensile Pregrasp for Ungraspable Objects. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–10.

Yuanpei Chen, Chen Wang, Li Fei-Fei, and C Karen Liu. 2023a. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation. *arXiv preprint arXiv:2309.00987* (2023).

Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. 2018. Physics-based motion capture imitation with deep reinforcement learning. In *Proceedings of the 11th ACM SIGGRAPH Conference on Motion, Interaction and Games*. 1–10.

Alexander Clegg, Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. 2018. Learning to dress: Synthesizing human dressing motion via deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–10.

CMU. 2003. *CMU Graphics Lab Motion Capture Database*. http://mocap.cs.cmu.edu/

Zhiyang Dou, Xuelin Chen, Qingnan Fan, Taku Komura, and Wenping Wang. 2023. C·ase: Learning conditional adversarial skill embeddings for physics-based characters. In *SIGGRAPH Asia 2023 Conference Papers*. 1–11.

Zicong Fan, Omid Taheri, Dimitrios Tzionas, Muhammed Kocabas, Manuel Kaufmann, Michael J Black, and Otmar Hilliges. 2023. ARCTIC: A dataset for dexterous bimanual hand-object manipulation. 12943–12954.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved training of Wasserstein GANs. *arXiv preprint arXiv:1704.00028* (2017).

Félix G. Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. 2020. Robust Motion In-Betweening. 39, 4 (2020).

Minsu Kim, Eunho Jung, and Yoonsang Lee. 2025. PhysicsFC: Learning User-Controlled Skills for a Physics-Based Football Player Controller. *arXiv preprint arXiv:2504.21216* (2025).

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

George Konidaris and Andrew Barto. 2009. Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in neural information processing systems* 22 (2009).

Paul G Kry and Dinesh K Pai. 2006. Interaction capture and synthesis. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 872–880.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems* 29 (2016).

Ariel Kwiatkowski, Eduardo Alvarado, Vicky Kalogeiton, C Karen Liu, Julien Pettré, Michiel van de Panne, and Marie-Paule Cani. 2022. A survey on reinforcement learning methods in character animation. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 613–639.

Youngwoon Lee, Joseph J Lim, Anima Anandkumar, and Yuke Zhu. 2021. Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization. *arXiv preprint arXiv:2111.07999* (2021).

Jae Hyun Lim and Jong Chul Ye. 2017. Geometric GAN. *arXiv preprint arXiv:1705.02894* (2017).

Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel van de Panne. 2020. Character controllers using motion VAEs. *ACM Trans. Graph.* 39, 4, Article 40 (2020).

C Karen Liu. 2008. Synthesis of interactive hand manipulation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 163–171.

C Karen Liu. 2009. Dextrous manipulation from a grasping pose. In *ACM SIGGRAPH 2009 papers*. 1–6.

Libin Liu and Jessica Hodgins. 2017. Learning to schedule control fragments for physics-based characters using deep q-learning. *ACM Transactions on Graphics (TOG)* 36, 3 (2017), 1–14.

Libin Liu and Jessica Hodgins. 2018. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–14.

Siqi Liu, Guy Lever, Zhe Wang, Josh Merel, SM Ali Eslami, Daniel Hennes, Wojciech M Czarnecki, Yuval Tassa, Shayegan Omidshafiei, Abbas Abdolmaleki, et al. 2022. From motor control to team play in simulated humanoid football. *Science Robotics* 7, 69 (2022), eabo0235.

Chaoyi Luo, Pengbin Tang, Yuqi Ma, and Dongjin Huang. 2024. Learning to Play Guitar with Robotic Hands. In *Computer Graphics Forum*, Vol. 43. Wiley Online Library, e15166.

Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. 2019. AMASS: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*. 5442–5451.

Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. 2021. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. arXiv:2108.10470 [cs.RO]

Josh Merel, Yuval Tassa, Dhruva TB, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. 2017. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201* (2017).

Gyeongsik Moon, Takaaki Shiratori, and Shunsuke Saito. 2025. Expressive whole-body 3D gaussian avatar. In *European Conference on Computer Vision*. Springer, 19–35.

Igor Mordatch, Zoran Popović, and Emanuel Todorov. 2012. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*. 137–144.

Liang Pan, Jingbo Wang, Buzhen Huang, Junyu Zhang, Haofan Wang, Xu Tang, and Yangang Wang. 2024. Synthesizing physically plausible human motions in 3d scenes. In *2024 International Conference on 3D Vision (3DV)*. IEEE, 1498–1507.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. 2018a. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)* 37, 4 (2018), 1–14.

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *Acm transactions on graphics (tog)* 36, 4 (2017), 1–13.

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. 2019. MCP: Learning Composable Hierarchical Control with Multiplicative Compositional Policies. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 3681–3692.

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. 2022. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions On Graphics (TOG)* 41, 4 (2022), 1–17.

Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. 2018b. Sfv: Reinforcement learning of physical skills from videos. *ACM Transactions On Graphics (TOG)* 37, 6 (2018), 1–14.

Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (ToG)* 40, 4 (2021), 1–20.

Nancy S Pollard and Victor Brian Zordan. 2005. Physically based grasping control from example. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. 311–318.

Avinash Ranganath, Pei Xu, Ioannis Karamouzas, and Victor Zordan. 2019. Low dimensional motor skill learning using coactivation. In *Proceedings of the 12th ACM SIGGRAPH Conference on Motion, Interaction and Games*. 1–10.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG]

Jingyu Shi, Rahul Jain, Hyungjun Doh, Ryo Suzuki, and Karthik Ramani. 2023. An HCI-Centric Survey and Taxonomy of Human-Generative-AI Interactions. *arXiv preprint arXiv:2310.07127* (2023).

Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. 2020. Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 54–1.

Omid Taheri, Nima Ghorbani, Michael J Black, and Dimitrios Tzionas. 2020. GRAB: A dataset of whole-body human grasping of objects. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*. Springer, 581–600.

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel Mankowitz, and Shie Mannor. 2017. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 31.

Chen Tessler, Yoni Kasten, Yunrong Guo, Shie Mannor, Gal Chechik, and Xue Bin Peng. 2023. Calm: Conditional adversarial latent models for directable virtual characters. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–9.

Hado P Van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. 2016. Learning values across many orders of magnitude. *Advances in neural information processing systems* 29 (2016).

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. Feudal networks for hierarchical reinforcement learning. In *International conference on machine learning*. PMLR, 3540–3549.

Chen Wang, Haochen Shi, Weizhuo Wang, Ruohan Zhang, Li Fei-Fei, and C Karen Liu. 2024c. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788* (2024).

Jiashun Wang, Jessica Hodgins, and Jungdam Won. 2024a. Strategy and skill learning for physics-based table tennis animation. In *ACM SIGGRAPH 2024 Conference Papers*. 1–11.

Ruocheng Wang, Pei Xu, Haochen Shi, Elizabeth Schumann, and C. Karen Liu. 2024d. FürElise: Capturing and Physically Synthesizing Hand Motions of Piano Performance. In *SIGGRAPH Asia 2024 Conference Papers (SA '24)*. Association for Computing Machinery, New York, NY, USA, Article 77, 11 pages.

Wenjia Wang, Liang Pan, Zhiyang Dou, Zhouyingcheng Liao, Yuke Lou, Lei Yang, Jingbo Wang, and Taku Komura. 2024b. SIMS: Simulating Human-Scene Interactions with Real World Script Planning. *arXiv preprint arXiv:2411.19921* (2024).

Yinhuai Wang, Jing Lin, Ailing Zeng, Zhengyi Luo, Jian Zhang, and Lei Zhang. 2023. Physhoi: Physics-based imitation of dynamic human-object interaction. *arXiv preprint arXiv:2312.04393* (2023).

Yangang Wang, Jianyuan Min, Jianjie Zhang, Yebin Liu, Feng Xu, Qionghai Dai, and Jinxiang Chai. 2013. Video-based hand manipulation capture through composite motion control. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–14.

Yufu Wang, Ziyun Wang, Lingjie Liu, and Kostas Daniilidis. 2025. TRAM: Global Trajectory and Motion of 3D Humans from in-the-wild Videos. In *European Conference on Computer Vision*. Springer, 467–487.

Yinhuai Wang, Qihan Zhao, Runyi Yu, Ailing Zeng, Jing Lin, Zhengyi Luo, Hok Wai Tsui, Jiwen Yu, Xiu Li, Qifeng Chen, et al. 2024e. Skillmimic: Learning reusable basketball skills from demonstrations. *arXiv preprint arXiv:2408.15270* (2024).

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2020. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics* 39, 4 (2020), 33–1.

Zhen Wu, Jiaman Li, and C Karen Liu. 2024. Human-object interaction from human-level instructions. *arXiv preprint arXiv:2406.17840* (2024).

Zeqi Xiao, Tai Wang, Jingbo Wang, Jinkun Cao, Wenwei Zhang, Bo Dai, Dahua Lin, and Jiangmiao Pang. 2023. Unified human-scene interaction via prompted chain-of-contacts. *arXiv preprint arXiv:2309.07918* (2023).

Zhaoming Xie, Hung Yu Ling, Nam Hee Kim, and Michiel van de Panne. 2020. Allsteps: curriculum-driven learning of stepping stone skills. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 213–224.

Zhaoming Xie, Jonathan Tseng, Sebastian Starke, Michiel van de Panne, and C Karen Liu. 2023. Hierarchical planning and control for box loco-manipulation. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 6, 3 (2023), 1–18.

Pei Xu and Ioannis Karamouzas. 2021. A GAN-Like Approach for Physics-Based Imitation Learning and Interactive Character Control. *Proc. of the ACM on Computer Graphics and Interactive Techniques* 4, 3 (2021).

Pei Xu, Xiumin Shang, Victor Zordan, and Ioannis Karamouzas. 2023a. Composite Motion Learning with Task Control. *ACM Transactions on Graphics* 42, 4 (2023).

Pei Xu and Ruocheng Wang. 2024. Synchronize Dual Hands for Physics-Based Dexterous Guitar Playing. In *SIGGRAPH Asia 2024 Conference Papers (SA '24)*. Association for Computing Machinery, New York, NY, USA, Article 143, 11 pages.

Pei Xu, Kaixiang Xie, Sheldon Andrews, Paul G. Kry, Michael Neff, Ioannis Karamouzas, and Victor Zordan. 2023b. AdaptNet: Policy Adaptation for Physics-Based Character Control. *ACM Transactions on Graphics* 42, 6 (2023).

Zeshi Yang, Kangkang Yin, and Libin Liu. 2022. Learning to use chopsticks in diverse gripping styles. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–17.

Heyuan Yao, Zhenhua Song, Baoquan Chen, and Libin Liu. 2022. ControlVAE: Model-Based Learning of Generative Controllers for Physics-Based Characters. *ACM Trans. Graph.* 41, 6, Article 183 (2022).

Heyuan Yao, Zhenhua Song, Yuyang Zhou, Tenglong Ao, Baoquan Chen, and Libin Liu. 2023. MoConVQ: Unified Physics-Based Motion Control via Scalable Discrete Representations. *arXiv preprint arXiv:2310.10198* (2023).

Yuting Ye and C Karen Liu. 2012. Synthesis of detailed hand manipulations using contact sampling. *ACM Transactions on Graphics (ToG)* 31, 4 (2012), 1–10.

Zhiqi Yin, Zeshi Yang, Michiel Van De Panne, and KangKang Yin. 2021. Discovering diverse athletic jumping strategies. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–17.

Kevin Zakka, Philipp Wu, Laura Smith, Nimrod Gileadi, Taylor Howell, Xue Bin Peng, Sumeet Singh, Yuval Tassa, Pete Florence, Andy Zeng, and Pieter Abbeel. 2023. RoboPianist: Dexterous Piano Playing with Deep Reinforcement Learning. In *Conference on Robot Learning (CoRL)*.

Haotian Zhang, Ye Yuan, Viktor Makoviychuk, Yunrong Guo, Sanja Fidler, Xue Bin Peng, and Kayvon Fatahalian. 2023. Learning Physically Simulated Tennis Skills from Broadcast Videos. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–14.

Wenping Zhao, Jianjie Zhang, Jianyuan Min, and Jinxiang Chai. 2013. Robust realtime physics-based motion control for human grasping. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–12.

Qingxu Zhu, He Zhang, Mengting Lan, and Lei Han. 2023. Neural categorical priors for physics-based character control. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–16.

A Implementation Details

We perform control over the seven phases (see Figure 2) during basketball games through six primitive policies, where defending behaviors are considered as special stances during locomotion and are integrated with a locomotion policy, plus two intermediate policies of gathering for shooting and passing respectively. All the primitive policies are trained under a multi-objective learning framework for physics-based character control using reinforcement learning [Xu et al. 2023a]. The system architecture for primitive policy learning is shown in Figure 4. For motion imitation, we utilize partially observable (hands-only or body-only) reference motions collected from different sources, and combine the usage of basketball-playing motions with non-basketball-playing motions for locomotion during dribbling. In contrast to previous work [Wang et al. 2024e] using detailed full-body motions with ball trajectories for skill learning, our approach employs a GAN-like architecture [Xu and Karamouzas 2021] and enables motion imitation from unstructured sources to achieve more flexible results of motion synthesis without needing pre-collected or generated [Liu and Hodgins 2018] ball trajectories for reference. For adaptation to primitive policies during the training for the policy transition, we use the fine-tuning approach from AdaptNet [Xu et al. 2023b]. This approach adapts a pretrained policy through latent space manipulation, and allows introducing new context input during adaptation.

We use PPO [Schulman et al. 2017] as the backbone reinforcement learning algorithm and take the Adam optimizer [Kingma and Ba 2017] to perform network optimization for policy training. The hyperparameters used for policy training are listed in Table S1 and the network structures are shown in Figure S1. The optimization function for each primitive policy learning can be written as

$$\max \mathbb{E}_t \left[\sum\nolimits_{\kappa} w_k \bar{A}_{t,k} \log \pi(\mathbf{a}_t | \mathbf{s}_t) \right] \tag{S1}$$

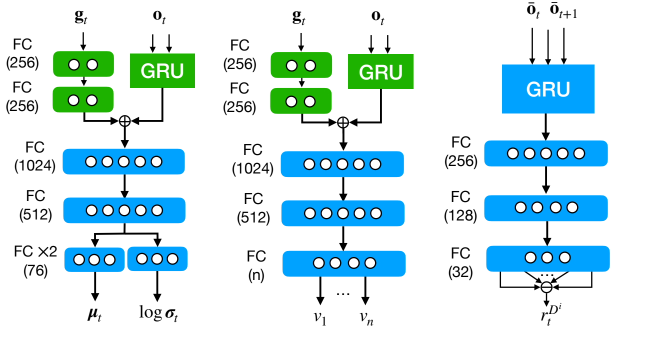
where $\bar{A}_{t,k}$ is the standardized advantage that is estimated according to the achieved reward of each objective k, and w_k is an associated weight. The number of objectives and the associated weights differ depending on the given task, which are summarized in Table S2.

Following previous literature [Xu and Karamouzas 2021], the imitation-related reward is obtained through a discriminator ensemble D using hinge loss [Lim and Ye 2017]:

$$r_t^{\text{imit},i}(\bar{\mathbf{o}}_t^i, \bar{\mathbf{o}}_{t+1}^i) = \frac{1}{N} \sum_{n=1}^{N} \textsc{Clip}\left(D_n^i(\bar{\mathbf{o}}_t^i, \bar{\mathbf{o}}_{t+1}^i), -1, 1 \right), \tag{S2}$$

where the subscript i indicates different imitation objectives. Most of our policies have two imitation objectives for hand-only and body imitation (see Table S2). Therefore, instead of using a full observation to the character, we use $\bar{\mathbf{o}}_t^i$ and $\bar{\mathbf{o}}_{t+1}^i$ to represent partially observable states of the character for the corresponding discriminator. We employ an ensemble of N discriminators ($N = 32$ in our implementation) for each imitation objective and the discriminator is trained to minimize the hinge loss with gradient penalty [Gulrajani et al. 2017]. We refer to the previous literature [Xu and Karamouzas 2021] for details.

For the value network, we employ a multi-head network structure and train it for the multiple objectives of each primitive policy. We employ the technique of PopArt [Van Hasselt et al. 2016] to perform



(a) Policy Network (b) Value Network (c) Discriminator Network

Fig. S1. Network structures where the input state $\mathbf{s}_t$ includes $\mathbf{g}_t$ as the goal state input and $\mathbf{o}_t$ as the observation to the character in the last two frames, and $\bar{\mathbf{o}}_t$ and $\bar{\mathbf{o}}_{t+1}$ represents the partial observation to the character in two consecutive frames. We use $\oplus$ denoting the add operator and $\ominus$ denoting the average operator. The state encoder consists of blocks in green, where a GRU encoder is employed for pose state encoding temporally, and a two-layer MLP encoder is employed for goal state encoding. The value network is a multi-head network whose output has a dimension of n depending on the number of objectives in a given task. We have $n = 4$ for the dribbling policy, 2 for passing and shooting policies, and 3 for the others (cf. Table S2).

Table S1. Hyperparameters. The number of simulated environments would be doubled or tripled during policy transition learning involving multiple primitive policies.

Parameter	**Value**
policy network learning rate	5×10^{-6}
critic network learning rate	1×10^{-4}
discriminator learning rate	1×10^{-5}
reward discount factor (γ)	0.95
GAE discount factor (λ)	0.95
surrogate clip range (ϵ)	0.2
gradient penalty coefficient (λ^{GP})	10
number of PPO workers (simulation instances)	512
PPO replay buffer size	4096
PPO batch size	256
PPO optimization epochs	5
discriminator replay buffer size	8192
discriminator batch size	512

value normalization on the multiple outputs of a value network. The normalized state value associated with the task reward is also used for intermediate policy learning (see Section 5) or for the preceding policy adaptation in the scene without the intermediate policy (e.g., catching to shooting and catching to passing).

Figure S2 shows the simulated character and the dimensions of the basketball court. The character has 57 body links and 26 controllable joints with 76 degrees of freedom. This results in a state space $\mathbf{s}_t \in \mathbb{R}^{2\times57\times13}$ including the character's position, orientation, and linear and angular velocities of each body link in the last two historical frames. For ball control, we only take into account the

Table S2. Objective weights for primitive policy training. A simple weight choice of 0.2, 0.1 and 0.7 (for body imitation, hand imitation, and goal-directed task, respectively) works in most cases. The listed weights are fine-tuned for fast task learning with minimal compromise in motion imitation performance. The rebounding policy use the same weights as the catching policy.

	Dribble	Shoot	Pass	Catch	Gather	Locomotion
Body imit.	0.2	0.4[†]	0.2[†]	0.08	0.2	0.25
Hand imit.	0.1	-	-	0.02	0.1	0.15
Task	0.7[*]	0.6	0.8	0.9	0.7	0.6

[*] dynamic weights for navigation and dribbling (see Section A.1).

[†] full-body with hand

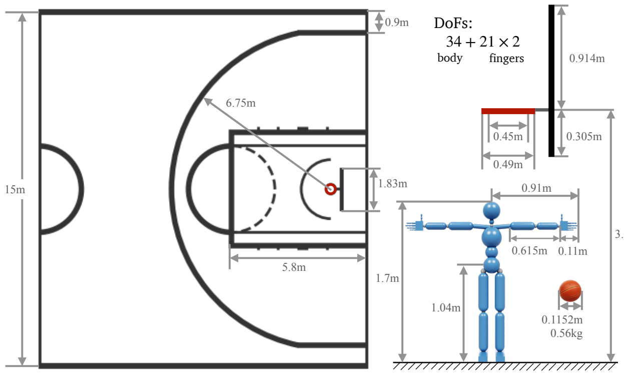


Fig. S2. Dimensions of the basketball court and character in our implementation. Our settings follow the standard of Olympic basketball games for women. The character has a normal height and intends to imitate our source motions collected from non-professional basketball players.

ball's latest dynamics without historical information and ignore its orientation given its round shape. This leads to a space of $\mathbb{R}^9$, including the ball's position, and linear and angular velocities, in the goal state vector $\mathbf{g}_t$, except for the locomotion policy which does not need ball state information. Depending on the given task, $\mathbf{g}_t$ could change to include additional information, for example, target velocity for locomotion, hoop positions relative to the character, pivoting foot, and dribbling state of the ball, which will be elaborated in the following subsections. All joints in our implementation are controlled through a PD servo. The policy network outputs the target posture to the PD servo and leads to an action space $\mathbf{a}_t \in \mathbb{R}^{76}$. We use IsaacGym [Makoviychuk et al. 2021] as our physics simulation engine. All policies run at 30Hz, while the simulation runs at 120Hz.

A.1 Dribbling Policy

Besides two imitation objectives, the dribbling policy has two goal-directed rewards for velocity-controlled navigation and dribbling, respectively.

The navigation reward is defined based on the error between the character's current horizontal velocity $\mathbf{v}$ and the target velocity

$\mathbf{v}_{\text{target}} \in \mathbb{R}^2$:

$$r_{\text{nav}} = \exp\left(-\frac{2}{\max\{1, ||\mathbf{v}_{\text{target}}||^2\}}||\mathbf{v} - \mathbf{v}_{\text{target}}||^2\right). \tag{S3}$$

The target velocity is given to the policy as part of the goal state in the form of velocity direction and magnitude. During training, the target velocity is generated randomly with a speed sampled in the range between 0 and 4m/s. To help train in-place dribbling motions, there is a chance around 20% to draw a zero target velocity.

For dribbling, we perform violation detection to check if there is any invalid contact between the ball and the character's body links other than the hands. To prevent consecutive dribbling performed in a row before the ball drops on the ground, we include an additional 0/1 variable c_{dribble} with the ball state to indicate if the ball currently should be dribbled or not. Along with the three goal state variables from the navigation objective, this leads to a goal state $\mathbf{g}_t^{\text{dribble}} \in \mathbb{R}^{13}$ in total for navigation while dribbling. The reward function for dribbling is defined as

$$r_{\text{dribble}} = 0.6r_{\text{hand}} + 0.4r_{\text{sp}} + 0.5I_{\text{dribble}}(0.2 + 0.8r_{\text{fingers}}). \tag{S4}$$

Specially, $r_{\text{dribble}} = -1$ if any violation is detected.

The term r_{hand}, as shown in Figure S3, measures the distance between the character's hand and the ball, encouraging the character to position their hands with palm direction facing toward the ball:

$$r_{\text{hand}} = \begin{cases} \exp\left(-2||\mathbf{p}_{\text{palm}}^{\kappa} + R_{\text{ball}}\mathbf{d}_{\text{plam}}^{\kappa} - \mathbf{p}_{\text{ball}}||\right) \\ \qquad\qquad \text{if } c_{\text{dribble}} = 1 \text{, i.e., the ball should be dribbled,} \\ \exp\left(-5||\min_{a>0}\mathbf{p}_{\text{palm}}^{\kappa} + a\mathbf{d}_{\text{plam}}^{\kappa} - \mathbf{p}_{\text{ball}}||^2\right) \quad \text{otherwise,} \end{cases} \tag{S5}$$

where $\mathbf{p}_{\text{palm}}^{\kappa}$ and $\mathbf{p}_{\text{ball}}$ are the positions of the palm and ball with $\kappa \in \{\text{left}, \text{right}\}$ to indicate the target hand, $\mathbf{d}_{\text{palm}}^{\kappa}$ is the facing direction of the palm, and R_{ball} is the radius of the ball. In the first case, when the ball should be dribbled, the reward encourages the palm to face the center of the ball while reaching for it, preventing dribbling with the back of the hand. In the second case, where the basketball has been dribbled and has not yet rebounded from the ground, the reward computes the projection of $\mathbf{p}_{\text{ball}}$ on the palm facing direction $\mathbf{d}_{\text{plam}}$. It only encourages the palm to face toward the ball rather than reaching the ball. The reward in the second case uses a squared error and is more tolerant than the one in the first case using an absolute error. It allows the palm to roughly face toward the ball, while the reward in the first case expects the palm to contact with the ball as closely as possible to generate human-like hand poses to interact with the ball. The value of a can be found by

$$a = \max\{0, (\mathbf{p}_{\text{ball}} - \mathbf{p}_{\text{palm}}) \cdot \mathbf{d}_{\text{plam}}\}. \tag{S6}$$

We decide κ by the nearest hand to the ball, i.e.

$$\kappa = \arg\min_{\kappa} ||\mathbf{p}_{\text{palm}}^{\kappa} - \mathbf{p}_{\text{ball}}^{\kappa}|| \tag{S7}$$

The term of r_{sp} in Eq. S4 measures the vertical speed of the ball and checks if the ball can rebound to at least the height of the pelvis.

$$r_{\text{sp}} = \min\{1, |v_{\text{ball}}/v_{\text{target}}|\} \tag{S8}$$

where v_{ball} is the current vertical speed of the ball. The target speed v_{target} is computed according to the current height of the character's pelvis link h_{pelvis}. It takes into account the restitution coefficient e between the ball and the ground to compute the rebound height if the ball is moving toward the ground vertically (i.e., $v_{\text{ball}} < 0$), or consider only the impact from gravity if the ball is moving upward:

$$v_{\text{target}} = \begin{cases} \sqrt{-2g(h_{\text{pelvis}} - h_{\text{ball}})} & \text{if } v_{\text{ball}} > 0, \\ \sqrt{-2g(h_{\text{pelvis}}/e^2 - h_{\text{ball}})} & \text{otherwise,} \end{cases} \tag{S9}$$

where $g = -9.81m/s^2$ is the gravitational acceleration and $e = 0.875$ is the restitution coefficient between the foot and ground. When $h_{\text{pelvis}} < h_{\text{ball}}$ or $h_{\text{pelvis}}/e^2 < h_{\text{ball}}$, we have $r_{\text{sp}} = 1$. The restitution coefficient is decided by measuring the bounce height of the ball, according to the basketball rules, in the physics simulator. It takes into account the simulation error and is slightly different from the physical quantity in the real world. In our experiments, we find that the introduction of r_{sp} is crucial for the policy to master continuous dribbling behaviors.

The term of $I_{\text{dribble}}r_{\text{fingers}}$ in Eq. S4 measures the finger pose when dribbling happens, where I_{dribble} is a 0/1 variable to indicate if a dribble happens. It is equal to 1 if the ball is pushed downward by a

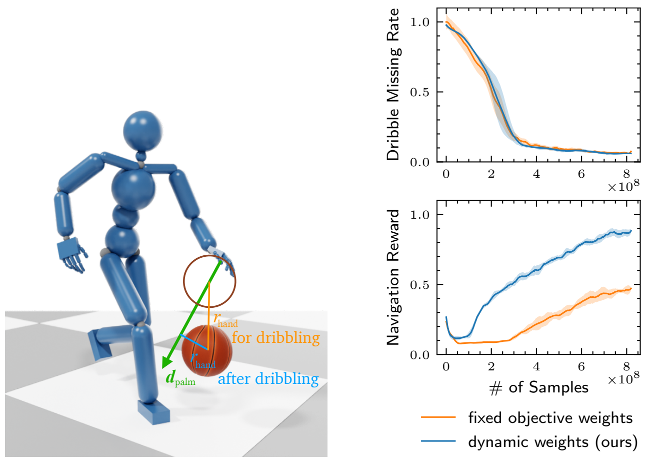


Fig. S3. Left: demonstrations of reward computation for ball dribbling before (orange) and after (blue) a dribble happens. Right: comparison of learning performance using dynamic task objective weights and that using fixed objective weights.

ALGORITHM S1: Ball Dribbling State Detection

```
if $v_{ball}^t - v_{ball}^{t-1} < \Delta V - 0.01$ and $v_{ball}^t < 0$ then
    $I_{\text{dribble}} = 1$ ;                        // A dribble happens.
    $c_{\text{dribble}} = 0$ ;        // Another dribble should not start.
else
    $I_{\text{dribble}} = 0$
if $v_{ball}^{t-1} < 0$ and $v_{ball}^t > 0$ then
    // Ball bounces up from the ground.
    // Count the dribble missing rate.
    if $c_{dribble} = 1$ then
        $n_{\text{dribble}}^- = n_{\text{dribble}}^- + 1$ ;          // Missing one dribble.
    else
        $n_{\text{dribble}}^+ = n_{\text{dribble}}^+ + 1$
    $c_{\text{dribble}} = 1$ ;              // Another dribble is allowed.
```

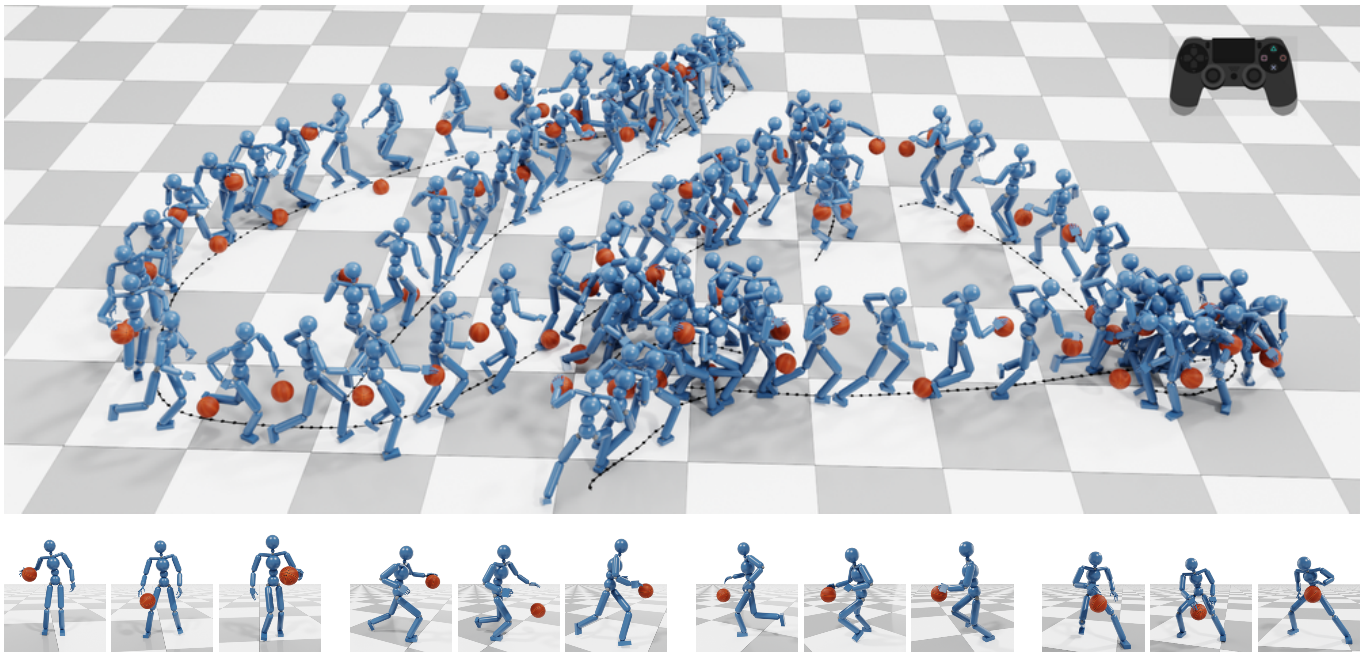


Fig. S4. Demonstrations of our dribbling policy trained using partially observable motion data. The dribbling policy is trained with unstructured motion data collected from multiple sources without ball trajectories. The character controlled by the policy can interact with the ball validly using hands and support interactive control by giving arbitrary target velocities with a valid speed in the range between 0m/s (inclusive) and 5m/s. Bottom: close-up pictures captured during the character walking, running and in-place dribbling.

hand. The definition of r_{fingers} can be written as

$$r_{\text{fingers}} = \exp\left(-10 \sum_{f \in \mathcal{F}^\kappa} |\mathbf{p}_{\text{f}}^\kappa - \mathbf{p}_{\text{ball}}| - R_{\text{ball}}\right) \tag{S10}$$

where $\mathcal{F}^\kappa$ is the set all fingertips plus the palm of the hand κ, and $\mathbf{p}_{\text{f}}^\kappa$ is the global position of a fingertip or the palm center in 3D space. This reward encourages the hand to dribble the ball with all fingertips in contact with the ball.

Instead of relying on accurate contact information, we use a simple heuristic rule to decide the ball's dribbling states, as shown in Algorithm S1, where $\Delta V = g/\text{FPS}$ is the velocity change of free-falling objects in the physics simulator, given that IsaacGym uses Euler's method for integration.

Since we do not have references of ball trajectories, the ball is initialized in front of the character randomly with a distance from 0.5 to 0.8m. If the character falls down, a termination reward of -25 will be given, and the simulation episode terminates. For invalid dribbling behaviors or violations, however, we adopt a soft termination strategy to re-initialize the position of the ball in front of the character and assign a penalty reward of -1 to the policy, while the simulation episode does not terminate. By such, we avoid the policy termination occurring too often at the beginning of training, which could prevent the policy from even learning to walk.

Additionally, we take a dynamic weighting strategy for the dribbling policy training. Since dribbling is much more difficult than locomotion and there is a potential conflict between the two tasks, to balance the two task objectives, a simple solution is to assign a larger weight in Eq. S1 to the dribbling task, given its higher priority. However, this approach can negatively impact the learning of locomotion. To address this issue, we set a lower weight to the locomotion objective initially to let the policy learn more about dribbling at the beginning of training, but increase it progressively as the success rate of dribbling (p_{dribble}) increases, namely, the dribble missing rate ($1 - p_{\text{dribble}}$) decreases. In our implementation, p_{dribble} is counted by moving average based on the policy's performance in training rollout (8 simulation steps with 512 parallel environments) with a fading coefficient of 0.9 on the past performance, i.e.

$$p_{\text{dribble}} \leftarrow 0.9 p_{\text{dribble}} + 0.1 \frac{n_{\text{dribble}}^+}{n_{\text{dribble}}^- + n_{\text{dribble}}^+} \tag{S11}$$

where n_{dribble}^- and n_{dribble}^+ are the number of counted successful and missed dribbles (cf. Algorithm S1). The weight for the locomotion objective is set dynamically and non-decreasing during training based on the dribble-missing rate through

$$w_{\text{nav}} \leftarrow 0.2 + 0.5 \max\left\{\exp\left(-10(1 - p_{\text{dribble}})\right), w_{\text{nav}}\right\} \tag{S12}$$

with an initial value of $w_{\text{nav}} = 0.2$, and the weight for the dribble objective is set by

$$w_{\text{dribble}} = 0.7 - w_{\text{nav}}. \tag{S13}$$

The two goal-directed tasks have a total weight of 0.7, and the total weight of the two motion imitation objectives is 0.3. In Figure S3, we compare the learning performance using our dynamic weight strategy and that using fixed weights (0.2 for navigation and 0.5 for dribbling, i.e., the initial weights for the dynamic weight strategy). From the figure, we can see that as the dribble missing rate decreases,

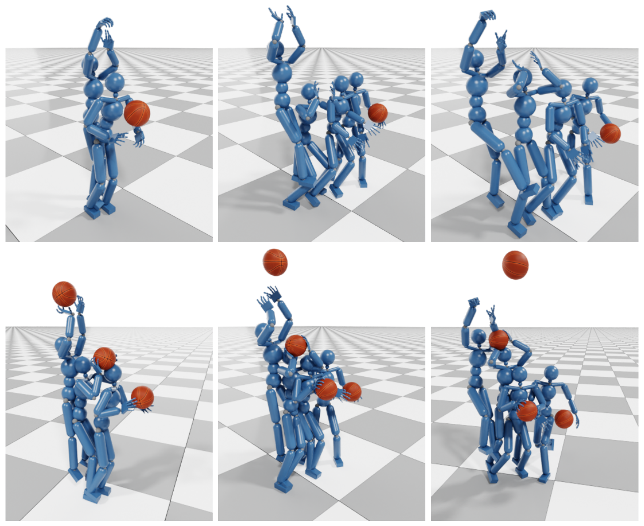


Fig. S5. Top: reference motions of jump shooting without ball trajectories but a manually decided initial position of the ball. Bottom: close look of the learned shooting motions.

there is no performance loss when we reduce the weight of dribbling (blue). However, the increase in the associated locomotion weight makes the learning of navigation more effective.

A.2 Shooting Policy

For the shooting policy, we have a full-body (including hands) imitation objective and a goal-directed reward that measures the shot accuracy and ball-holding performance before the ball is shot out. Besides the ball state, the shooting policy also takes the horizontal position of the hoop relative to the character as the goal state. This leads to a goal state vector $\mathbf{g}_t^{\text{shoot}} \in \mathbb{R}^{11}$.

Due to missing ball trajectory reference, we manually set the ball's initial position near the character's hand in each of the three reference motions that we have for policy learning, as shown in Figure S5. This initialization scheme makes the ball easily catchable for the character through motion imitation and avoids unexpected penetrations between the ball and character caused by random initialization. Nevertheless, such an initialization scheme lets the trained policy see only a very limited set of the ball states and thus makes it sensitive to the initial pose of the ball during execution. Each training episode will end after 60 frames (2s) and terminate if the ball has not been released with two hands holding the ball after 40 frames.

The reward for shooting encourages the character to first hold and then lift the ball using two hands before shooting the ball out, like a human player would do for a jumping shoot:

$$r_{\text{shoot}} = \begin{cases} 0.5 r_{\text{hands}} + I_{\text{up}} r_{\text{hold}} & \text{before the ball is released,} \\ r_{\text{release}} + \exp(-0.25 ||\hat{\mathbf{p}}_{\text{ball}} - \mathbf{p}_{\text{hoop}}||) & \text{otherwise.} \end{cases} \tag{S14}$$

Similar to the first case in Eq. S5, r_{hands} encourages the hand to approach the ball with the palm facing toward the ball. However, while dribbling uses only one hand, r_{hands} here takes into account the two hands at the same time:

$$r_{\text{hands}} = \exp\left(\frac{-5}{|\kappa|} \sum_{\kappa} |\mathbf{p}_{\text{palm}}^{\kappa} + R_{\text{ball}} \mathbf{d}_{\text{palm}}^{\kappa} - \mathbf{p}_{\text{ball}}|\right) \tag{S15}$$

where $\kappa \in \{\text{left}, \text{right}\}$ and $|\kappa| = 2$. $I_{\text{up}} = 1$ if the ball is lifted up compared to previous frames or 0 otherwise. r_{hold} is similar to Eq. S10, but measures the performance of two hands holding the ball simultaneously. The definition of r_{hold} can be written as

$$r_{\text{hold}} = \exp\left(-20 \sum_{\kappa} \sum_{f \in \mathcal{F}^{\kappa}} |\mathbf{p}_{\text{f}}^{\kappa} - \mathbf{p}_{\text{ball}} - R_{\text{ball}}|\right). \tag{S16}$$

By multiplying r_{hold} by I_{up}, we expect that the character could lift the ball with fingers and palms in close contact with the ball before shooting it out.

In the second case of Eq. S14, r_{release} is defined as

$$r_{\text{release}} = h_{\text{release}} / h_{\text{hoop}} \tag{S17}$$

where h_{release} is the height of the ball when it is held by the hand just before being released. This reward term encourages the character to shoot the ball from a higher position rather than directly throwing the ball out. In the last term, $\hat{\mathbf{p}}_{\text{ball}}$ is the estimated minimal distance between the ball flying trajectory and the hoop on the horizontal plane at the hoop height. Figure S6 shows how the points are chosen given the ball's projectile decided by its current linear velocity. We estimate $\hat{\mathbf{p}}_{\text{ball}}$ under the assumption that the ball is only affected by gravity, and then correct the estimation based on the actual simulated result of the ball state.

Though the initial posture is fixed, the character will be randomly put in a ring area around the hoop with an inner radius of 2.5m and an outer radius of 7.5m, as shown in Figure 8. The initial facing direction will also be adjusted with some randomness to ensure that the character can face the hoop roughly (with a maximal error of 20°). As discussed in the experiment section, the shooting policy is sensitive to the initial posture and ball state, but can perform shots with high accuracy in the effective area.

We do not perform violation detection when training the vanilla, primitive policy of shooting. The major violation that could happen during the phases from ball catching to jump shooting is traveling, which barely occurs with effective full-body imitation of the reference motions starting with a jump-ready pose. However, during adaptation learning for policy transition from dribbling or catching, traveling could happen very frequently, given the character dynamic states. In that case, we introduce an additional variable in the goal state vector to indicate the pivoting foot for the policy to avoid traveling. The details of traveling detection will be elaborated in Section A.8.

A.3 Passing Policy

The passing task is very similar to shooting with the major difference in (1) the goal state and (2) the falling points chosen for reward computation. The goal state for passing includes the ball state and the target position of passing in the 3D space relative to the character's root link. This leads to a goal state $\mathbf{g}_t^{\text{pass}} \in \mathbb{R}^{12}$. Similar to the training of the shooting policy, an additional indicator of the pivoting foot will be introduced into the goal state during

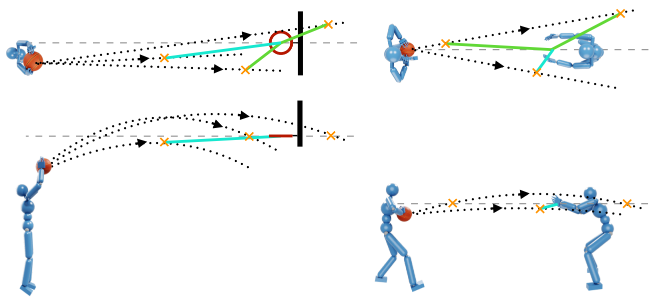


Fig. S6. The chosen points (yellow cross markers) on the ball projectile used for reward computation for shooting (left) and passing (right) tasks. We compute the 2D distance from the falling points on the horizontal plane of the target (blue lines) or the 3D distance (cyan) from the top of the projectile if the projectile is not high enough to reach the target height. For the shooting task, we consider the falling point only. For the passing task, we consider the two possible points where the projectile intersects with the plane at the target height.

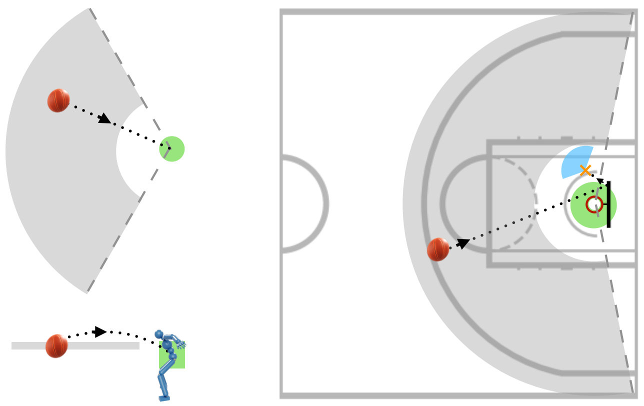


Fig. S7. Ball initialization for the training of the catching (left) and rebounding (right) policies. The gray areas indicate the area where the ball will be launched randomly. A falling point in the green area will be chosen randomly to decide the initial velocity of the ball. The falling point is chosen around the catching player while training the catching policy. During rebounding policy training, the character will be randomly placed at the front direction (blue area) of the ball's actual falling point (yellow cross marker) after colliding, if there is, with the hoop and board.

adaptation learning for policy transition, but not in the pre-training phase of the primitive policy.

While during reward computation, the shooting task only takes the valid falling point where the ball drops from a higher position above the hoop, the passing task allows the ball to reach the target position from any direction (see Figure S6). Ideally, we prefer to find the closest position on the ball's projectile to the target position for reward computation. However, this will lead to an optimization problem involving a cubic equation, which is not easy to solve. Therefore, instead of finding the closest point in the 3D space, we adopt a similar strategy of reward computation for the shooting policy, but consider the closest one from the two possible intersection points of the ball's projectile and the horizontal plane of the target position for reward computation.

During training, the position of the passing target is chosen randomly in a half ring area in front of the character with an inner radius of 2.5m and an outer radius of 7.5m at a random height between 0.8 and 1.1m. During deployment for interactive control, the chest position of the target character is chosen as the target position of passing.

A.4 Catching Policy

The catching policy has the same goal state with the passing policy, but the indicator variable for the pivoting foot is always introduced to prevent traveling after ball catching. The goal state, therefore, is $\mathbf{g}_t^{\text{catch}} \in \mathbb{R}^{13}$. We refer to Section A.8 for details of traveling and pivoting foot detection.

The catching reward is similar to the ball-holding reward used for the shooting policy:

$$r_{\text{catch}} = 0.5 r'_{\text{hands}} + r_{\text{hold}} - I_{\text{traveling}} \tag{S18}$$

where $I_{\text{traveling}}$ is a 0/1 indicator for foot traveling. The term r'_{hands} is an extend to r_{hands} in Eq. S15:

$$r'_{\text{hands}} = 0.3 \exp(-e_{\text{hands}}) + 0.7 r_{\text{hands}} \tag{S19}$$

where $e_{\text{hands}} = \frac{1}{|\kappa|} \sum_\kappa |\mathbf{p}^\kappa_{\text{palm}} + R_{\text{ball}} \mathbf{d}^\kappa_{\text{palm}} - \mathbf{p}_{\text{ball}}|$. As the distance from the hands/fingers to the ball may vary a lot during the catching process, we take the trick from previous literature [Xu and Wang 2024], and use the sum of two exponential functions for reward computation. This trick prevents it from falling into the saturation range of an exponential function when the error is large, while keeping the reward function sensitive when facing small errors.

To improve the training efficiency, the ball during training is initialized to fly towards the character roughly, such that the character can try to catch the ball effectively. Figure S7 shows the area in gray from which the initial position of the ball is randomly drawn. It is a ring arc area in front of the character with an inner radius of 2.5m and an outer radius of 7.5m. The angle of the arc is 120°. The initial height is randomly chosen from 0.8 to 1.1m. The target position is randomly sampled from a circular area about the character with a radius of 0.5m (green in the figure). The target height is -0.1m to 0.5m around the root height of the character. We consider the ball's traveling time as a random number between 0.3s (10 frames) and 0.83s (50 frames). The initial linear velocity of the ball is then decided by the sampled initial position, target position, and traveling time.

A.5 Rebounding Policy

The rebounding policy is similar to catching. They are trained using exactly the same reference motions and goal-directed reward in Eq. S18. However, while the catching policy faces the scenario where the ball flies in the air barrier-freely, the rebounding policy faces the scenario where the ball would collide with the hoop and/or board and change its trajectory. To enable the policy to detect the potential impacts from the collision, we use the goal state from the shooting policy, which includes the hoop's position besides the ball's state.

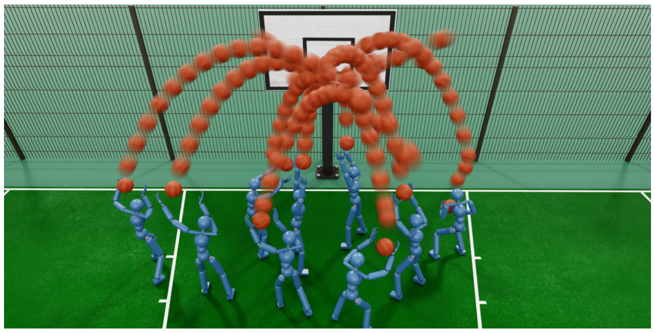


Fig. S8. Character controlled by our rebounding policy can perform rebounding in arbitrary positions around the hoop near the falling point.

Similarly to the training of the catching policy, we also introduce the pivoting foot indicator in the goal state to prevent foot traveling.

Figure S7 shows how the ball is initialized for rebounding during training. To emulate the pre-rebounding states of the ball, we run additional simulation environments in parallel to collect ball trajectories with falling points near the hoop. We procedurally shoot the ball from the common shooting area (the gray area in the figure) at a random height between 1.5m and 3m to the hoop range, as shown by the green area in the figure, excluding the center of the hoop. After collecting the falling trajectories of the ball, which may collide with the hoop and/or board or just free-fall, we initialize the character in front of the falling point (the blue area). The facing direction of the character is roughly toward the ball's horizontal direction of velocity at a randomly sampled angle within 120° and a distance within 1m. We also constrain the ball's maximal launching speed to 3m/s horizontally and angle to 45°. Since the whole flying trajectory of the ball may be quite long, taking at most 2 seconds, we cut the trajectory, and extract the ball pose in the last 20th frame (2/3 seconds) before the ball drops at the hoop height as the ball pose initialized together with the character for policy training.

Instead of mimicking certain pre-collected motions to catch the ball with a fixed set of dynamic states, our trained policy can control the character to perform rebounding in any valid position around the ball's falling point given an arbitrary initial state. Figure S8 exhibits some results of our rebounding policy. The character under control can, in advance, predict the falling point of the ball after colliding with the hoop and/or board, and adjust its pose for better rebounding. The introduction of foot traveling detection prevents the character from randomly moving after catching the ball. We refer to our supplementary video for the animated results.

A.6 Gathering Policy

The gathering policy is not a completely independent policy. It is introduced as the intermediate policy for transition purposes between dribbling and shooting, as described in Section 5. Besides the shooting off the dribbling, we also adopt the same setup to train a gathering policy for passing off the dribbling in our implementation. Except for the common part of the ball state, the goal state differs depending on the succeeding policy. In the shooting-off-the-dribbling task, the goal state has the hoop position like the shooting policy but with one more indicator for the pivoting foot to prevent traveling. In the passing-off-the-dribbling task, the goal state is the pivoting foot indicator plus the position of the passing target used for the passing policy.

The reward function defined for the gathering policy is similar to the reward for the shooting policy, but focuses only on the ball-catching behaviors without the ball-releasing reward. Meanwhile, for policy transition purposes (cf. Eq. 1), we utilize the state value function from the adapted shooting policy (i.e. π^{+}_{shoot}) to guide the character reaching a state preferred by the shooting policy. A similar approach is used for the passing-off-the-dribbling task while adapting the passing policy. We refer to Eq. 1 for the full reward definition, where

$$r_{\text{pose}} = 0.3 r'_{\text{hands}} + 0.2 r_{\text{orient}} + r_{\text{hold}} - I_{\text{traveling}}. \tag{S20}$$

This reward is similar to the reward function for the catching policy (Eq. S18) but has one more term r_{orient} measuring the facing direction error of the character. r_{orient} is computed by

$$r_{\text{orient}} = \exp\left(-4\left(\frac{|\arccos\theta|}{\pi}\right)^3\right) \tag{S21}$$

where θ is the angle between the character's pelvis (root) heading direction and the horizontal direction from the character to the hoop in the shooting-off-the-dribbling task and to the target position in the passing-off-the-dribbling task.

During the gathering policy training, a dribbling policy runs parallelly in another batch of simulation environments to produce random initial states of the character and ball for the gathering policy. While taking the produced local dynamics of the character and ball, the character controlled by the gathering policy will be teleported to a random position near the hoop in the same range as the shooting policy training to perform ball gathering for shooting in the shooting-off-the-dribbling task. Based on the estimated state value, we pass states of the gathering character and ball to the succeeding policy for further adaptation. We will terminate the episode without penalty if the ball is out of control (flying away or rolling/bounding on the ground) within 40 frames (1.3s). A termination penalty (a reward of −25) will be given only when the character falls down.

A.7 Defending and Locomotion Policy

We combine the locomotion and defending behavior into one policy. The character is expected to perform locomotion when a horizontal target velocity ($\mathbf{v}_{\text{target}} \in \mathbb{R}^2$) is given, or to stretch his arms up, down or horizontally in place according to the given command when no target velocity is given, as shown in Figure S9. This leads to a goal state $\mathbf{g}_t^{\text{loco}} \in \mathbb{R}^4$ where three of the elements represent the target velocity in the form of direction and magnitude, like that for the dribbling policy, and an additional one for defensive pose indication. The reward function is defined as:

$$r_{\text{loco}} = \begin{cases} r_{\text{nav}} & \text{if } \mathbf{v}_{\text{target}} \text{ is given,} \\ 1 + 0.2\exp(-||\mathbf{v}||^2) + 0.8 r_{\text{style}} & \text{otherwise,} \end{cases} \tag{S22}$$

where r_{nav} is the navigation reward from Eq. S3. When no target velocity is given, we want the character to stay in place by using the 2nd reward case to minimize its horizontal linear velocity $\mathbf{v}$.

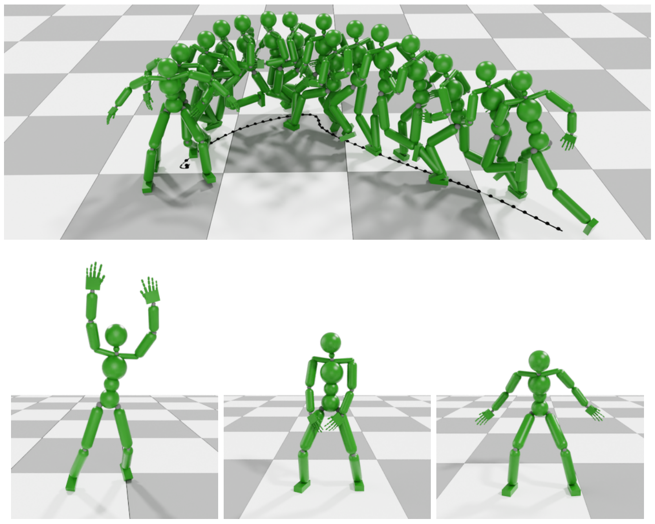


Fig. S9. Top: controllable locomotion with defensive stances. Bottom: Close look at defensive behaviors corresponding to different arm stretching poses. From left to right: block, screen, and defensive stance.

Instead of using a conditional discriminator [Dou et al. 2023; Tessler et al. 2023] for defensive stance control we rely on the reward term r_{style} to guide the stance styles for the policy. r_{style} is defined mainly according to to the arms' stretching state:

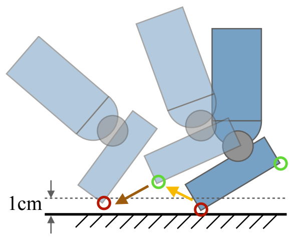
$$r_{\text{style}} = \begin{cases} r_{\text{block}} & \text{if blocking command is given,} \\ r_{\text{screen}} & \text{if screening command is given,} \\ r_{\text{defend}} & \text{otherwise.} \end{cases} \tag{S23}$$

where

$$\begin{aligned} r_{\text{block}} &= 0.25 \max_{\kappa} \min\{1, \theta_{\text{l}}^{\kappa}/0.16\pi\} \\ &\quad + 0.75 \max_{\kappa} \min\{1, (\theta_{\text{u}}^{\kappa} + 0.212\pi)/0.376\pi\}, \\ r_{\text{screen}} &= 0.25 \min\{1, (0.5 - ||\mathbf{p}_{\text{palm}}^{\text{left}} - \mathbf{p}_{\text{palm}}^{\text{right}}||)/0.3\} \\ &\quad + 0.75 \min_{i \in l,u} \min_{\kappa} \min\{1, (0.4\pi - \theta_{\text{i}}^{\kappa})/0.8\pi\}, \\ r_{\text{defend}} &= \min_{\kappa} \min\{1, (0.5\pi - |\theta_{\text{u}}^{\kappa}|)/0.334\pi)\}. \end{aligned} \tag{S24}$$

Those reward measures the vertical angle for the upper ($\theta_{\text{u}}^{\kappa}$) and lower ($\theta_{\text{l}}^{\kappa}$) arms given $\kappa \in \{\text{left, right}\}$, and encourages the character to reach the desired arm stretching pose. For blocking, the character obtains the maximal reward of 1 if any of the upper arms and lower arms raise up more than 30° at the same time. For screening, the character obtains the maximal reward when the two hands are close enough with a distance less than 0.2m, and all lower and upper arms put down with a vertical angle more than 70°. In the third case, the character obtains the maximal reward if all upper arms are stretched horizontally with a vertical angle within the range of ±30°.

Because during training, the target velocity and arm stretching commands are generated randomly with a higher proportion of locomotion rather than staying in place, to balance the learning of defending behaviors and locomotion, we start the training with all environments generating only zero target velocity for defensive behavior learning, and then progressively increase the proportion of environments accepting dynamic target velocity, which also could be zero. This approach allows the character to master the simple defending poses first before learning locomotion. Additionally, we adopt an adaptive sampling strategy to ensure the simulation steps spent on the learning of three different defending poses are equal roughly. During interactive control, the command of blocking and screening is given by the user, while the default command for standing in place is the defensive stance.



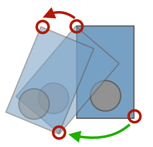


Fig. S10. Demonstrations of foot traveling detection. Left: a foot with points contacting with ground (red circle) lifting up and dropping down would lead to traveling if the foot is a pivoting foot, or will let the other foot become the pivoting foot if the pivoting foot is undecided (two feet contacted with the ground at the same time previously when the ball is caught). Right: foot movement with at least one contacting point not changing is allowed (green arrow), but the traveling will be triggered if all contacting points move by a distance above 0.01m (red arrow) from the initial contacting points.

A.8 Foot Traveling Detection

We perform traveling detection according to the pivoting foot rules in real basketball games. In practice, however, traveling is mainly decided visually. It allows the player to rotate around a foot (the pivoting foot) rather than requiring that foot to be fixed on the ground strictly while holding the ball. In our implementation, we introduce additional tolerance to avoid a too rigorous detection of traveling. Given that each foot in our character model is shaped as a cuboid, we consider the four vertices on the bottom side of the foot cube for contact detection. A contact is recorded if any of the four vertices of a foot is less than 0.01m above the ground. Traveling will be decided when the ball is holding and if (1) a pivot foot falls back on the ground (from a non-contacting state to a contacting state) with the ball held by the hands, or (2) a pivot foot moves with *all* recorded contact points traveling more than 0.01m horizontally. The tolerance given in the second rule is crucial for allowing the character to perform pivoting where the body spins with one foot contacting the ground. Figure S10 gives two examples of traveling detection. In the right example, the movement along the green arrow is allowed, since one contacting point (the left top corner of the foot) is fixed and the movement is actually a rotation around that contacting point. However, the movement along the red arrow, even after moving along the green arrow first, is considered traveling, as we perform traveling detection by measuring the movement of the contacting points from the initial contact positions.

The pivoting foot rule in the basketball game treats the foot that first touches the ground as the pivoting foot when or after the ball is caught. The rule allows the player to pivot on either foot if two feet are contacting the ground when catching the ball or touching the

ALGORITHM S2: Pivoting Foot and Traveling Detection

```
Input  : $p_{t-1} \in \{-1, 0, 1, 2\}$: previous pivoting foot state, $p_0 = -1$.
         $\mathcal{P}, m_f, d_f, T_f^t, c_f^t$ from FootMovementDetection for
         $f \in$ {left foot, right foot}.
Output : traveling $\in$ {true, false}: traveling state.
         $p_t$: updated pivoting foot state.
1  traveling = false
2  if ball is not held by any hand then
3      $p_t = -1$ ;                    // Undefined pivoting foot.
4      $\mathcal{P} \leftarrow \emptyset$ ;        // Clear recorded contacting states.
5      $c_f^t$ = false, $\mathcal{T}_f^t = -4 \quad \forall f$
6  else
7      $p_t = p_{t-1}$
8      if $p_t = -1$ then
           // No foot-ground contact was detected.
9          if $c^t_{left\ foot}$ and $c^t_{right\ foot}$ then
               // Two feet contact ground simultaneously.
                  Either foot can be the pivoting foot.
10             $p_t = 2$
11         else if $c^t_{left\ foot}$ then
12             $p_t = 0$ ;                     // Left foot pivots.
13         else if $c^t_{right\ foot}$ then
14             $p_t = 1$ ;                    // Right foot pivots.
15     else if $p_t == 2$ then
           // Pivoting foot is not decided.
           // Two feet contacted ground simultaneously
              after ball catching.
16         if $d_f$ for any $f \in$ {left foot, right foot} then
17             traveling = true
18         else if $m_f$ for all $f \in$ {left foot, right foot} then
19             traveling = true
20         else if $c^t_{left\ foot}$ is false or ($c^t_{right\ foot}$ is true and $m_{left\ foot}$) then
               // Left foot moves and right foot becomes the
                  pivoting foot
21             $p_t = 1$
22             traveling = $m_{\text{right foot}}$
23         else if $c^t_{right\ foot}$ is false or ($c^t_{left\ foot}$ is true and $m_{right\ foot}$)
            then
               // Right foot moves and left foot becomes the
                  pivoting foot
24             $p_t = 0$
25             traveling = $m_{\text{left foot}}$
26     else if $p_t == 0$ then
           // Pivoting foot is the left foot.
27         traveling = $m_{\text{left foot}}$ or $d_{\text{left foot}}$
28     else if $p_t == 1$ then
           // Pivoting foot is the right foot.
29         traveling = $m_{\text{right foot}}$ or $d_{\text{right foot}}$
       if $c_f$ and $T_f > t - 4$ for all $f \in$ {left foot, right foot} then
           // 4-frame tolerance to detect if two foot
              contact ground simultaneously.
30         $p_t = 2$
```

ground simultaneously after catching the ball. In the latter case, once a foot lifts up, the other foot will become the pivoting foot. However, it is too rigorous to perform the detection of two-foot pivoting at the time scale of 30Hz. Therefore, we introduce an additional 4-frame tolerance. If two feet contact the ground within 4 frames after the catch, we consider that the two feet contact simultaneously, and either of the two feet can be treated as the pivoting foot (i.e. the undecided state of the pivoting foot).

For reference, we elaborate on our algorithm for pivoting foot and traveling detection in Algorithms S2 and S3.

ALGORITHM S3: Foot Movement Detection

```
Input  : $t$: the current timestep.
         $\mathcal{P}_{t-1}$: an array or table recording the contacting position of
         each foot link vertex $i$. $\mathcal{P} \leftarrow \emptyset$ during initialization.
         $T_f^{t-1}$ for $f \in$ {left foot, right foot}: recording of the
         timestep at which the contact is detected for each foot
         link $f$. $T_f = -4$ during initialization for a 4-frame tolerance
         to detect two-foot pivoting.
         $\mathbf{p}_i \in \mathbb{R}^3$: the current Cartesian coordinate of each foot link
         vertex $i$ in the global space. The third element, $\mathbf{p}_i[2]$,
         represents the vertical height above the ground.
         $c_f^{t-1} \in$ {true, false}: the previous foot-ground contacting
         state.
Output : $c_f^t \in$ {true, false}: the ground contacting state of each foot.
         $m_f \in$ {true, false}: the movement state of each foot.
         $d_f \in$ {true, false}: the dropping state of each foot.
         $\mathcal{P}_t$: updated positions of contacting vertices.
         $T_f^t$: updated timestep recording of contacting foot.
1  for each contacting vertex $i$ in $C$ do
2      if $|\mathcal{P}[i][0] - \mathbf{p}_i[0]| > 0.01$ or $|\mathcal{P}[i][1] - \mathbf{p}_i[1]| > 0.01$ then
3          vertex $i$ moves
   end
4  for each $f \in$ {left foot, right foot} do
5      $c_f$ = false, $d_f$ = false, $m_f$ = false
6      if $\mathbf{p}_i[2] < 0.01$ for any vertex $i$ belongs to the foot $f$ then
7          $c_f^t$ = true ;          // Foot $f$ is contacting ground.
8          if $c_f^{t-1}$ is false then
9              $d_f$ = true ;            // Foot $f$ drops on ground.
10         if vertex $i$ moves for all $i$ belongs to the foot $f$ then
11             $m_f$ = true ;                     // Foot $f$ moves.
   end
12 $\mathcal{P}_t \leftarrow \mathcal{P}_{t-1}$, $T_f^t \leftarrow T_f^{t-1}$
13 for each $f \in$ {left foot, right foot} do
14     for each vertex $i$ belongs to the foot $f$ do
15         if $i$ not in $\mathcal{P}$ and $\mathbf{p}_i[2] < 0.01$ then
               // Record the first-time contacting
                  coordinate for the vertex.
16             $\mathcal{P}_t[i] = \mathbf{p}_i$
17             if $T_f^t < 0$ then
                   // Record the first-time contacting
                      timestep for the foot.
18                 $T_f^t = t$
       end
   end
```

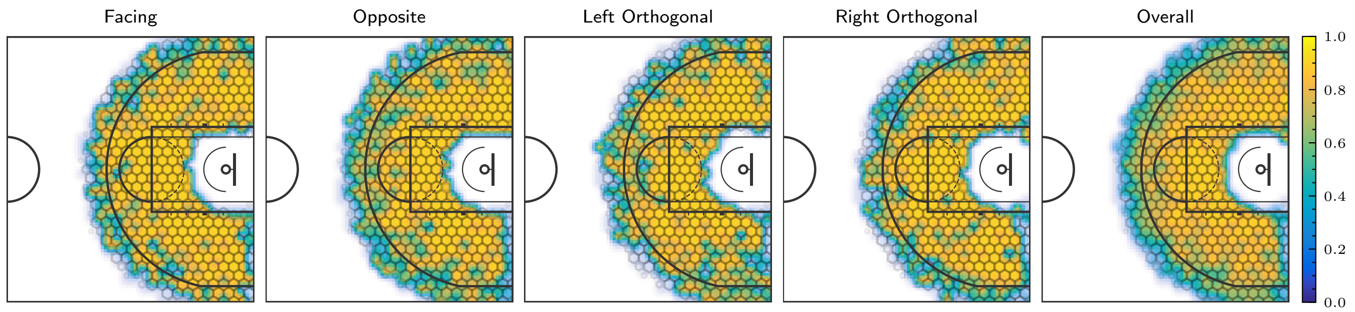


Fig. S11. Heatmap of the shot percentage of our system when the character dribbles at different approaching directions towards the hoop.

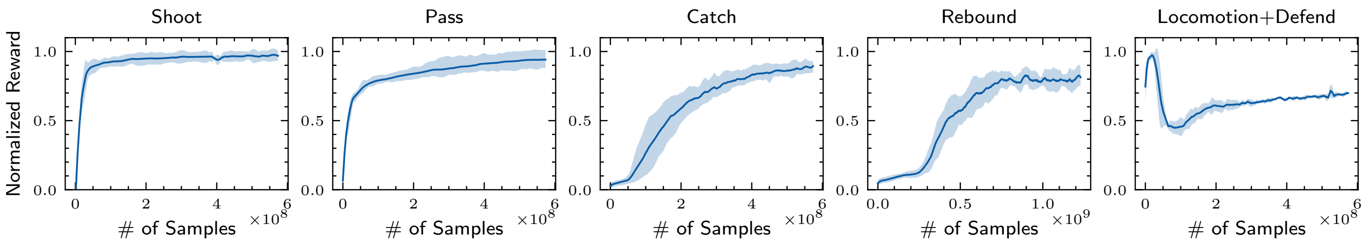


Fig. S12. Curves of the primitive policy training performance. The shaded ranges show the performance over three training trials.

B Transition Policy Training and Composing

While the primitive policies can be trained independently at the same time, our presented training scheme for policy transition (Type B and C in Figure 3) requires the preceding, succeeding, and intermediate (if there is) policies to be trained together to learn the transition. In our implementation, we run 512 simulation environments to train each primitive policy. During the training for policy transition, we run 512-by-n environments at the same time where $n = 2$ for mutual adaptation and $n = 3$ for the scheme using intermediate policy. Specially, for the rebounding policy, an additional batch of 512 environments run in parallel to collect ball trajectories from which the the pre-rebounding ball state is extracted to initialize the ball for the character to rebound (cf. Section A.5). We perform policy composition for the case where the transition should be done automatically but the transitional states between two policies are ill-defined, like the dribbling to shooting/passing transition through a gathering policy (Type C transitions), and the catching to shooting transition (Type B transitions). During interactive control, for example, we will call for the composed catching-to-shooting policy if catching and shooting commands are given at the same time. If the command of catching and shooting is given sequentially, the system will perform catching using the catching-passing adapted policy first, and then, after receiving the shooting command, call the shooting-off-the-dribbling policy as the default executor of shooting.

C Additional Results

Figure S11 shows additional visualization results of the shot percentage of the policy trained by our system. This figure is the visualization of the number reported in Table 2. In Figure S12, we show the learning performance of the primitive policies, in terms of the task reward. The performance of the dribbling policy is shown in Figure S3. For the locomotion+defend case, the policy is trained for in-place defensive stances initially, and the ratio of locomotion training is increased gradually as the training goes on (cf. Section A.7). This leads to a performance drop at the early training stage with the introduction of locomotion, as shown in the right-most subplot in the figure.

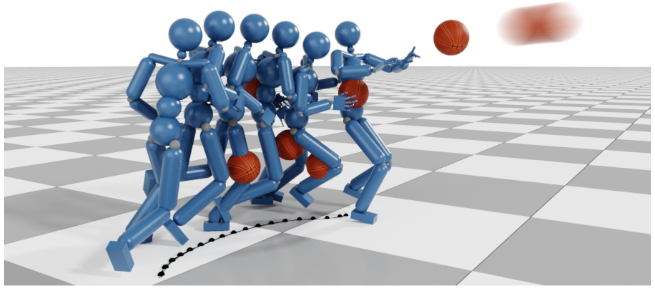


Fig. S13. A demonstration of passing off the dribbling. Similar to the shooting-off-the-dribbling example, the introduced intermediate, gathering policy can adjust the body pose according to the passing target's position for seamless transition to passing.