Supplementary for Learning Agile Skills via Adversarial Imitation of Rough Partial Demonstrations

A Training Details

A.1 Training Parameters

The learning networks and algorithm are implemented in PyTorch 1.10 with CUDA 11.3. Adam is used as the optimizer for the policy and value function with an adaptive learning rate with a KL divergence target of 0.01. The optimizer for the discriminator is SGD for LSGAN and RMSprop for WASABI. The discount factor γ is set to 0.99, the clip range ϵ is set to 0.2, and the entropy coefficient α is set to 0.01. The policy runs at 50 Hz. All training is done by collecting experiences from 4096 uncorrelated instances of the simulator in parallel. Most of the experiments are executed on the cluster of Max Planck Institute for Intelligent Systems with NVIDIA A100 and Tesla V100 GPUs. In this setting, one run with 5000 iterations with the specified compute settings and devices completes within 2 hours. The information is summarized in Table S1.

Table S1: Training parameters

Parameter	Symbol	Value
step time seconds	–	0.02
max episode time seconds	–	20
max iterations	–	5000
steps per iteration	–	24
policy learning epochs	–	5
policy mini-batches	–	4
KL divergence target	–	0.01
discount factor	γ	0.99
clip range	ϵ	0.2
entropy coefficient	α	0.01
discriminator weight decay	wd^{D}	0.001
discriminator momentum	–	0.05
discriminator gradient penalty coefficient	–	5.0
discriminator learning epochs	–	1
discriminator mini-batches	–	80
Wasserstein loss weight	w^{D}	0.5
gradient penalty weight	w^{GP}	5.0
parallel training environments	–	4096
number of seeds	–	5
approximate training hours	–	2
policy learning rate	lr^{π}	grid-searched
discriminator learning rate	lr^{D}	grid-searched
discriminator observation horizon	H	grid-searched
imitation reward relative importance	w^{I}	grid-searched

Note that policy learning rate, discriminator learning rate, discriminator observation horizon, and imitation reward relative importance are grid-searched and optimized in different tasks. We report the value used for evaluation in Sec. 4 in Table S2.

A.2 Network Architecture

The network architecture is detailed in Table S3, where H denotes the discriminator observation horizon.

Table S2: Task-specific parameters

Section	Task	Method	lr^{π}	lr^{D}	H	w^{I}
Sec. 4.1 (Sim)	SOLOBACKFLIP	LSGAN	1.0×10^{-6}	1.0×10^{-4}	2	0.8
		WASABI	1.0×10^{-7}	1.0×10^{-7}	16	4.0
Sec. 4.2 (Sim)	SOLOLEAP	LSGAN	1.0×10^{-4}	1.0×10^{-4}	2	0.8
		WASABI	1.0×10^{-7}	5.0×10^{-8}	2	8.0
	SOLOWAVE	LSGAN	1.0×10^{-6}	1.0×10^{-4}	4	0.8
		WASABI	1.0×10^{-6}	5.0×10^{-8}	4	0.8
	SOLOSTANDUP	LSGAN	1.0×10^{-4}	1.0×10^{-4}	2	0.8
		WASABI	1.0×10^{-7}	1.0×10^{-7}	4	4.0
	SOLOBACKFLIP	LSGAN	1.0×10^{-6}	1.0×10^{-4}	2	0.8
		WASABI	1.0×10^{-7}	1.0×10^{-7}	16	4.0
	SOLOSTANDUP†	LSGAN	5.0×10^{-5}	5.0×10^{-5}	8	0.8
		WASABI	1.0×10^{-7}	5.0×10^{-7}	2	4.0
	SOLOSTANDUP*	LSGAN	1.0×10^{-4}	1.0×10^{-5}	8	0.8
		WASABI	5.0×10^{-7}	1.0×10^{-7}	2	4.0
	SOLOBACKFLIP†	LSGAN	5.0×10^{-5}	1.0×10^{-5}	8	0.8
		WASABI	1.0×10^{-7}	5.0×10^{-8}	2	4.0
	SOLOBACKFLIP*	LSGAN	1.0×10^{-4}	1.0×10^{-5}	8	0.8
		WASABI	1.0×10^{-7}	5.0×10^{-8}	2	0.8
Sec. 4.3 (Real)	SOLOLEAP	WASABI	1.0×10^{-3}	5.0×10^{-7}	2	0.1
	SOLOWAVE	WASABI	1.0×10^{-3}	5.0×10^{-7}	2	0.1
	SOLOBACKFLIP	WASABI	1.0×10^{-3}	5.0×10^{-7}	2	0.4

Table S3: Network architecture

Network	Symbol	Type	Shape	Activation
policy	π	MLP	$68, 128, 128, 128, 8$	Exponential Linear Unit (ELU)
value function	V	MLP	$68, 128, 128, 128, 1$	Exponential Linear Unit (ELU)
discriminator	D	MLP	$10H, 512, 256, 1$	Rectified Linear Unit (ReLU)

A.3 Domain Randomization

Two types of domain randomization techniques are applied during training to improve policy performance when transferring from simulation to the real system.

On the one hand, the base mass of the parallel training instances is perturbed with an additional weight $m' \sim \mathcal{U}(-0.5, 1.0)$, where $\mathcal{U}$ denotes uniform distribution. On the other hand, random pushing is also applied every 15 seconds on the robot base by forcing its horizontal linear velocity to be set randomly within $v_{xy} \sim \mathcal{U}(-0.5, 0.5)$.

B Model Representation

B.1 Discriminator Observation

Table S4 lists the extracted features sent to the discriminator.

For discriminator observation horizon $H > 1$, the entries are concatenated to a vector of size $10H$. The resulting features are then normalized and used as inputs to the discriminator network.

B.2 Policy Observation and Action Space

In our work, we use a universal set of states as the policy observations for all tasks. It has 68 dimensions and consists of the same set of state measurements of two consecutive steps as detailed in

Table S4: Discriminator observation space

Entry	Symbol	Dimensions
base linear velocity	v	0:3
base angular velocity	ω	3:6
projected gravity	g	6:9
base height	z	9:10

Table S5: Policy observation space

Entry	Symbol	Dimensions	noise level b
base linear velocity	v	0:3	0.2
base angular velocity	ω	3:6	0.05
projected gravity	g	6:9	0.05
base height	z	9:10	0.01
joint positions	q	10:18	0.01
joint velocities	$\dot{q}$	18:26	0.75
last actions	a'	26:34	0.0

Table S5. The observation space is composed of base linear and angular velocities v, ω in the robot frame, measurement of the gravity vector in the robot frame g, base height z, joint positions q and velocities $\dot{q}$, and the most recent actions a'.

The noise level b denotes the artificial noise added during training to increase the policy robustness. Note again that the policy receives the observation collection for two consecutive steps. This is not necessarily optimal for all the tasks as less information would suffice for easier tasks. For simplicity, we use this fixed set of observations for all experiments.

The action space is of 8 dimensions and encodes the target joint position for each of the 8 actuators. The PD gains are set to 5.0 and 0.1, respectively.

C Regularization Reward Functions

The regularization reward functions contain only regularization terms whose formulation is detailed below. A universal set of involved hyperparameters is used across different motions and is summarized in Table S6.

C.1 Action Rate

$$r_{ar} = w_{ar}\|a' - a\|_2^2, \tag{S1}$$

where w_{ar} denotes the weight of the action rate reward, a' and a denote the previous and current actions.

C.2 Joint Acceleration

$$r_{q_a} = w_{q_a}\left\|\frac{\dot{q}' - \dot{q}}{\Delta t}\right\|_2^2, \tag{S2}$$

where w_{q_a} denotes the weight of the joint acceleration reward, $\dot{q}'$ and $\dot{q}$ denote the previous and current joint velocity, Δt denotes the step time interval.

Table S6: Regularization reward hyperparameters

Hyperparameter	w_{ar}	w_{q_a}	w_{q_T}	$w_{\dot{\phi}}$	$w_{\dot{\psi}}$	$w_{\dot{y}}$
Value	-0.005	-1.25×10^{-8}	-1.25×10^{-6}	-0.001	-0.001	-0.001

C.3 Joint Torque

$$r_{q_T} = w_{q_T} \|T\|_2^2, \tag{S3}$$

where w_{q_T} denotes the weight of the joint torque reward, T denotes the joint torques.

C.4 Angular Velocity x

$$r_{\dot{\phi}} = w_{\dot{\phi}} \left\|\dot{\phi}\right\|_2^2, \tag{S4}$$

where $w_{\dot{\phi}}$ denotes the weight of the angular velocity x reward, $\dot{\phi}$ denotes the base roll rate.

C.5 Angular Velocity z

$$r_{\dot{\psi}} = w_{\dot{\psi}} \left\|\dot{\psi}\right\|_2^2, \tag{S5}$$

where $w_{\dot{\psi}}$ denotes the weight of the angular velocity z reward, $\dot{\psi}$ denotes the base yaw rate.

C.6 Linear Velocity y

$$r_{\dot{y}} = w_{\dot{y}} \|\dot{y}\|_2^2, \tag{S6}$$

where $w_{\dot{y}}$ denotes the weight of the linear velocity y reward, $\dot{y}$ denotes the base lateral velocity.

D Method Comparison

In this section, we highlight the connections and differences between our work and the AMP work [1], in particular, how the LSGAN formulation in our setting relates to AMP.

D.1 AMP Adapted for Task Learning Termed as LSGAN

It is acknowledged that both WASABI and AMP utilize generative adversarial learning methods to learn motions from reference demonstrations. However, from a high-level view, WASABI aims to solve fundamentally different tasks as opposed to AMP.

Note that AMP itself does not target task reward learning. In AMP, the GAN model is employed to shape the styles of a learning agent while performing some other tasks, which are relatively straightforward and motivated by additional task reward functions (e.g. moving forward, reaching a target). This can be viewed as learning the regularization of motions. Indeed, we can adapt AMP to enable it to directly learn complex tasks by providing the desired motions as the reference and removing the original task reward. In addition, to alleviate the mode collapse issue as described in Sec. 3.2, the capability of the discriminator is extended to encompass longer observation horizons. With these adaptations, AMP is referred to as LSGAN in our work.

Typically, AMP learns demonstrated motions with sufficient exploration enabled by reference state initialization (RSI), where the agent is initialized randomly along the reference trajectories. However, in the setting of legged skill learning from only base information, RSI is not applicable due to the missing joint reference. This makes the robot difficult to directly imitate highly dynamic motions. And as shown in Table 1, the robot fails to adopt complex skills (e.g. SOLOSTANDUP, SOLOBACKFLIP) with the LSGAN implementation.

D.2 WASABI Designed for Task Learning

In contrast, WASABI is proposed to learn task rewards directly for agile motions from limited data. This becomes especially meaningful for tasks where the reward function is challenging to design and a decent expert is not immediately accessible. WASABI provides solutions to cases where we want to quickly develop a complex skill for robot learning, allowing us to hand-hold a robot (or an object) without actuating it and to learn directly from the demonstrated trajectories.

E Handcrafted Task Reward

Experts trained on handcrafted task rewards are used as a baseline to prove that WASABI is capable of extracting sensible task rewards. The policies learned using LSGAN and WASABI use 1000 sampled trajectories generated by the experts in simulation as reference.

E.1 Upright Stand for SOLOSTANDUP

The robot is encouraged to stand up by rewarding the pitch angle of the base, the base height, and the standing on only the two hind legs.

$$r_\beta = w_{\theta_z}\theta_z + w_z z + w_g c_g, \tag{S7}$$

where θ_z denotes the pitch angle with respect to the global z-axis, $w_{\theta_z} = 1.0$ denotes its weight. z denotes the base height and $w_z = 3.0$ denotes its weight. c_g is a binary variable that takes 1 if the front legs have no contact with the ground, $w_g = 2.0$ denotes its weight.

E.2 Traversed Angle for SOLOBACKFLIP

The robot is encouraged to perform back-flipping by rewarding its traversed angle around the y-axis while in the air. This reward will be given only when the robot lands.

$$r_\theta = w_\theta \theta [\![s \in \mathcal{L}]\!], \tag{S8}$$

where $w_\theta = 5.0$ denotes the weight of the traversed angle reward, θ denotes the traversed angle around y-axis while in the air. $\mathcal{L}$ is the set of robot landing states, and $[\![\cdot]\!]$ is the Iverson bracket.

F Sim-to-Real Transfer

To deploy the learned policy on the real system, the following adaptations are made to reduce the sim-to-real gap.

F.1 Observation Space Adaptation

Generally, policies trained in simulation suffer from poor performance when transferred to the real system. One primary reason for such failures is the incorrectly modeled system dynamics. As a result, the state transitions observed in simulation may divert from reality. On account of this, instead of using two consecutive steps of the observation collection in Suppl. B.2, only the observation at the current time step is used. The resulting policy observation space has thus only 34 entries and is detailed in Table S5.

F.2 Training Hyperparameter Adaptation

Some training hyperparameters are further refined and specifically adapted for the deployment on the real system as detailed in Table S2.

F.3 Reward Adaptation

To generate stable joint actuation and achieve task-specific high performance, a feet air time regularization reward is introduced to motivate higher off-ground steps during robot movement.

$$r_{t_f} = w_{t_f} \sum_{i=1}^{4} t_{f_i} [\![f_i \in \mathcal{C}]\!], \tag{S9}$$

where w_{t_f} denotes the weight of the feet air time reward, t_{f_i} denotes the time foot i stays in the air. $\mathcal{C}$ is the set of foot states touching the ground, and $[\![\cdot]\!]$ is the Iverson bracket. The resulting task-specific regularization reward setting is provided in Table S7.

Table S7: Regularization reward adaptation

Task	w_{ar}	w_{q_a}	w_{q_T}	$w_{\dot{\phi}}$	$w_{\dot{\psi}}$	$w_{\dot{y}}$	w_{t_f}
SOLOLEAP	-0.01	-2.5×10^{-7}	-2.5×10^{-5}	-0.05	-0.05	-0.05	0.10
SOLOWAVE	-0.01	-2.5×10^{-7}	-2.5×10^{-5}	-0.01	-0.01	-0.01	0.05
SOLOBACKFLIP	-0.01	-2.5×10^{-7}	-1.0×10^{-5}	-0.02	-0.02	-0.02	0.00

Figure S1: Hand-held motion demonstration for SOLOBACKFLIP. Note that the robot joints are not actuated and only base information is recorded.

F.4 Policy Transfer

In our work, the learned policies for SOLOLEAP, SOLOWAVE, and SOLOBACKFLIP successfully transfer to the real system. However, during the deployment of the policy for SOLOSTANDUP from either the handcrafted expert or obtained by WASABI, the robot fails to maintain balance after it manages to stand up. As Solo 8 does not have Hip Abduction Adduction (HAA) joints, it could be challenging to adapt against unexpected base rolling accordingly, which happens frequently after standing up. This makes this task especially difficult. In addition, the sim-to-real gap could also be potentially encoded in the inaccurate modeling of the point contacts between the feet and the ground, which the robot requires to adapt constantly to maintain its attitude.

G Motions

G.1 Data Collection

The reference motions containing only the base information (as detailed in Sec. B.1) are performed by a human demonstrator and recorded using a Vicon motion capture system as illustrated in Figure S1. For each motion, 20 trajectories are recorded and used as the reference motion dataset. The number of frames in the recorded trajectories for each motion is detailed in Table S8.

G.2 Motion Details

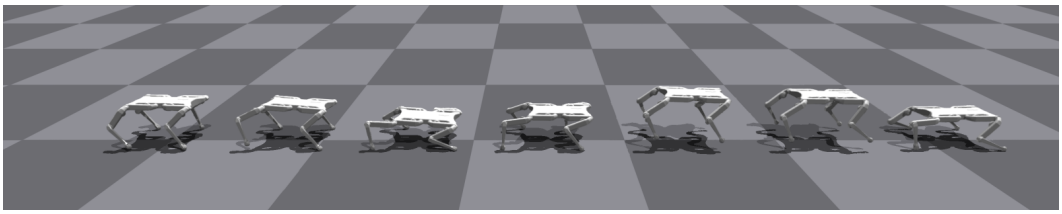
We provide sequences of the respective motions that we learn in this work in Fig. S2.

H Dynamic Time Warping Evaluation

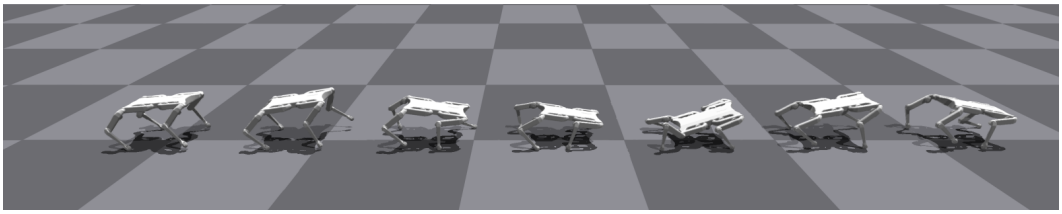
We make use of an imitation reward as outlined in Sec. 3.1 for learning policies from rough reference trajectories. Since the reference trajectories might not be completely achievable, either in terms of time consistency or the actual sequence of states that are infeasible, a simple evaluation metric such as the direct L_2 distance between policy and reference trajectories is not applicable. Furthermore, a learned policy might replicate the demonstrated motion perfectly in a certain sub-sequence, however, in the absence of proper alignment and synchronization, it would incur a large penalty in terms of a simple distance metric.

Table S8: Number of frames per recorded trajectory

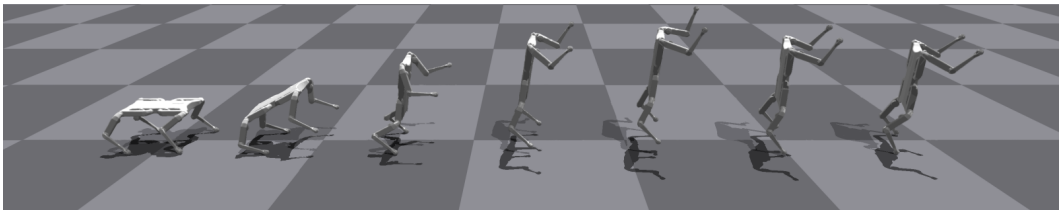
	SOLOLEAP	SOLOWAVE	SOLOSTANDUP	SOLOBACKFLIP
Frames	130	130	100	60



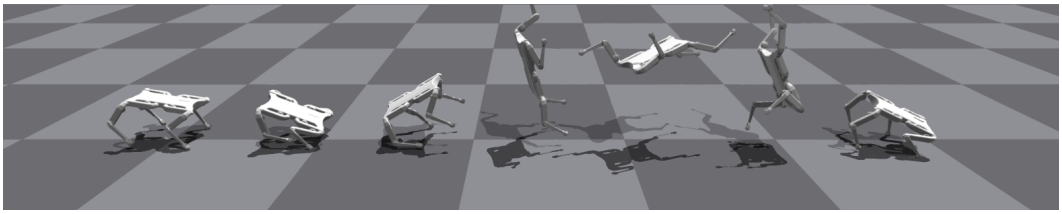
(a) SOLOLEAP



(b) SOLOWAVE



(c) SOLOSTANDUP



(d) SOLOBACKFLIP

Figure S2: Task motion sequences induced by the learned policies in simulation.

To account for the potential misalignment in the policy vs. reference trajectories, the policy trajectories need to be synchronized to the reference. Dynamic Time Warping (DTW) is an algorithm that computes an alignment between two sequences such that a distance measure between the elements of the sequences is minimized, under a set of matching constraints. This can be computed efficiently via dynamic programming, an example of such a matching is given for our case in Fig. 1.

For the distance measure between the matched elements of the sequences we use the standard L_2 distance, and for the matching computation, we use the 'dtw' Python package [2] with the Mori asymmetric step pattern [3] and open-ended matching. The asymmetric step pattern constrains the type of element matches that can happen between the trajectories and also makes it possible for some elements to be skipped, while open-ended matching allows for ends of trajectories to be matched to an earlier time point which is necessary to account for time shift. The resulting measure is denoted with d^{DTW}.

As clarified in Sec. 4, we calculate the expectation

$$\mathbb{E}\left[d^{\mathrm{DTW}}(\Phi(\tau_\pi), \tau_{\mathcal{M}})\right] \tag{S10}$$

as the evaluation metric, where $\tau_\pi \sim d^\pi$ is a state trajectory drawn from a policy rollout distribution and $\tau_{\mathcal{M}} \sim d^{\mathcal{M}}$ denotes a reference motion from the dataset. With the same notation as in Sec. 3.2, the function Φ maps each state in the state trajectory of the policy to the reference observation space $\mathcal{O}$. In practice, we estimate this by drawing 20 policy rollouts, comparing each of the 20 rollouts with

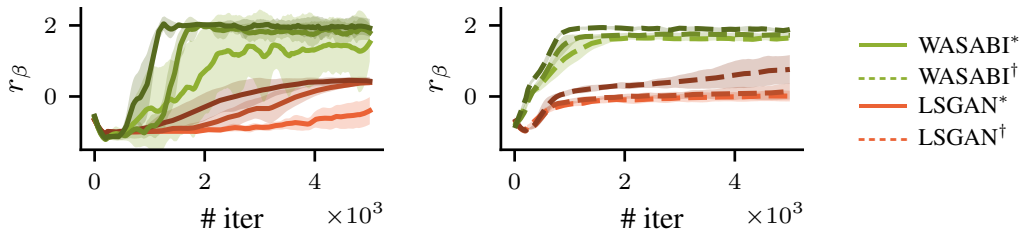


Figure S3: Performance of LSGAN (left) and WASABI (right) in terms of the handcrafted task reward for SOLOSTANDUP with different discriminator observation horizons (light $H = 2$, middle $H = 4$, dark $H = 8$). Solid lines indicate full information ($*$) and dashed lines indicate partial information (†).

20 collected reference trajectories, and taking the mean. Note that d^{DTW} is not comparable across different tasks, thus we provide a pure standing reference in Table 1.

I Ablation Studies

To prevent mode collapse, we extend the capability of the discriminator by allowing more than one state transition as input. In this section, we aim to investigate how the length of the discriminator horizon may affect the learning process of the desired behaviors. Here we provide an evaluation in terms of the handcrafted reward of policies learned with discriminator observations of horizon $H = 2, 4, 8$ with LSGAN and WASABI in SOLOSTANDUP. The result is depicted in Fig. S3.

Observe that a longer horizon tends to help the policy converge earlier and yield slightly better performance in terms of the handcrafted reward in both methods with either full or partial reference information, even if the LSGAN fails to learn the stand-up behavior. A similar pattern is also revealed in SOLOBACKFLIP, although it presents a weaker effect on policy convergence in comparison. The potential reason is that the policy learns to avoid producing state transitions that align with only a short sub-sequence of the reference motion. Such avoidance of mode collapse would prevent the policy from getting stuck at the local optima and thus increase the overall performance.

However, the benefit brought about by a longer discriminator observation horizon is not identified in tasks with hand-held reference motions where a direct performance metric is not applicable. Sometimes a longer horizon may even result in failure of learning the desired motion which is attainable with shorter horizons. An evaluation of performance improvement in terms of DTW over iterations reveals no clear pattern on how different discriminator observation horizons affect policy convergence. This may result from the time and state inconsistency in the hand-held reference motions which may already alleviate mode collapse to some extent.

J Extensions

In this section, we present extensions of WASABI by small modifications to the adversarial imitation learning framework proposed in our work. Supplementary videos for these extensions are available at https://sites.google.com/view/corl2022-wasabi/home.

J.1 Active Velocity Control

In our work, the hand-held reference motions are recorded by a human demonstrator. For locomotion tasks, if the demonstration speed is uniformly high in the reference dataset, the corresponding recorded velocity terms are also high. During the training of the policy, the robot will try to mimic the recorded data and thus operate at a high locomotion speed.

In contrast, if there is some variance in the velocity terms within the reference dataset, the policy will have larger freedom in choosing its locomotion velocity. For this reason, we take advantage of

the variance within the human demonstration and realize active velocity control by introducing a target-conditioned velocity tracking reward as follows

$$r_{\dot{x}} = w_{\dot{x}} e^{-\frac{(c_{\dot{x}} - \dot{x})^2}{\sigma_{\dot{x}}^2}}, \tag{S11}$$

where $w_{\dot{x}} = 0.5$ denotes the weight of the velocity tracking reward, $\dot{x}$ denotes the base longitudinal velocity, $c_{\dot{x}}$ denotes the velocity command on the same direction, which is also observed by the control policy. And $\sigma_{\dot{x}}^2 = 0.25$ denotes a temperature scale for the tracking error.

For SOLOLEAP and SOLOWAVE, we successfully enable active velocity control within the range $c_{\dot{x}} \in [0.1, 0.5]$ by utilizing the diversity of the demonstration speed within the reference dataset.

J.2 Single Flip in SOLOBACKFLIP

Using the same reference motion in SOLOBACKFLIP, active execution of a single flip is achieved by including an additional variable in the observation space of the control policy indicating whether the robot has finished a backflip. After the robot finishes a flip, the control policy stops receiving imitation signals from the discriminator by fixing a constant imitation reward output to prevent from learning a consecutive flip.

In the meanwhile, a stand-still reward term is imposed to regularize the joint configurations during landing and standing. The stand-still reward is formulated as follows

$$r_{q_0} = w_{q_0} \left\| q - q_0 \right\|_2^2, \tag{S12}$$

where $w_{q_0} = 0.01$ denotes the weight of the stand-still reward, q denotes the current joint positions and q_0 denotes the default joint positions when the robot stands.

References

[1] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa. AMP: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)*, 40(4):1–20, 2021.

[2] T. Giorgino. Computing and visualizing dynamic time warping alignments in r: the dtw package. *Journal of Statistical Software*, 31:1–24, 2009.

[3] A. Mori, S. Uchida, R. Kurazume, R.-i. Taniguchi, T. Hasegawa, and H. Sakoe. Early recognition and prediction of gestures. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 560–563. IEEE, 2006.