

MaskAdapt: Learning Flexible Motion Adaptation via Mask-Invariant Prior for Physics-Based Characters

Supplementary Material

A. Implementation Details

Base policies were trained for 30K iterations using 4,096 parallel environments. Residual policies were trained for 20K iterations, with 4,096 environments for motion composition and 2,048 for partial tracking. For goal-driven tasks, training was extended to up to 35K iterations. All policies were optimized with Adam optimizers, using learning rates 1×10^{-5} , 1×10^{-4} , and 1×10^{-4} for the actor, critic, discriminator, respectively. Other hyperparameters are listed in Tab. 1 and 2. All experiments were conducted on NVIDIA RTX 4090 GPUs. Training the base policy took about one day, and the residual policy took roughly 16.5 hours for motion composition and 13.5 hours for partial tracking.

State and Action Representation. The state is represented as $s_t = (\mathbf{h}_t, \mathbf{p}_t, \mathbf{q}_t, \dot{\mathbf{p}}_t, \dot{\mathbf{q}}_t) \in \mathbb{R}^{328}$, where $\mathbf{h}_t \in \mathbb{R}^1$ represents the root height, $\mathbf{p}_t \in \mathbb{R}^{(J-1) \times 3}$ the joint positions excluding the root, $\mathbf{q}_t \in \mathbb{R}^{J \times 6}$ the joint rotations using 6D representation [36], and $\dot{\mathbf{p}}_t, \dot{\mathbf{q}}_t \in \mathbb{R}^{J \times 3}$ the joint linear and angular velocities, respectively, all normalized with respect to the character’s heading transform. Here, J is the total number of joints (in our case 22). The action space matches the number of DoFs, yielding a 63-dimensional vector.

Reward. For learning the base policy, the imitation reward follows [19]:

$$r_t = -\log(1 - D^\pi(s, s')). \quad (7)$$

For dynamic motion composition, we use the same formulation but condition the discriminator on the active body-part mask m :

$$r_t = -\log(1 - D^\psi(s, s' | m)). \quad (8)$$

For the partial tracking task, we combine an imitation reward with a tracking reward:

$$r_t^{\text{imit}} = -\log(1 - D^\psi(\phi_K(s), \phi_K(s'))), \quad (9)$$

$$r_t^{\text{track}} = \exp(-\|\hat{\mathbf{q}}_t - \mathbf{q}_t\|^2), \quad (10)$$

where ϕ_K extracts the state corresponding to the pre-designated body parts \mathcal{J}_K , and $\hat{\mathbf{q}}_t$ denotes the target joint rotations from the generated motion. The final reward is $r_t = w^{\text{imit}} r_t^{\text{imit}} + w^{\text{track}} r_t^{\text{track}}$ with $w^{\text{imit}} = w^{\text{track}} = 0.5$. The residual policy is optimized through multi-objective learning with PopArt normalization [29, 33] to stabilize learning in the tracking task.

Early Termination. The episode ends when the character falls, determined by any rigid body falling below a height

Parameters	Value
horizon length	32
number of environments	4096
minibatch size	4096
actor learning rate	1e-5
critic learning rate	1e-4
discriminator learning rate	1e-4
discount factor γ	0.99
GAE parameter τ	0.95
clip range ϵ	0.2
critic loss weight λ_V	5.0
gradient penalty weight λ_{GP}	5.0
MI loss weight λ_{MI}	1.0
Masking probability ρ	0.8

Table 1. Hyperparameters for learning base policy.

Parameters	Value
horizon length	32
number of environments	4096 / 2048
minibatch size	4096
actor learning rate	1e-5
critic learning rate	1e-4
discriminator learning rate	1e-4
discount factor γ	0.99
GAE parameter τ	0.95
clip range ϵ	0.2
critic loss weight λ_V	5.0
gradient penalty weight λ_{GP}	5.0
gain coefficient w_{gain}	<i>learned</i>

Table 2. Hyperparameters for learning residual policy.

threshold, except for end effectors that are allowed to contact the ground. For partial tracking, we additionally adopt the early termination strategy of [30], ending the episode when the average reward over a fixed time window becomes sufficiently low.

B. Goal-Driven Tasks

We describe three goal-driven tasks in detail. The reward function formulation primarily follows prior work [20, 33].

Target Location. In this task, the objective is to move the character to a target location \mathbf{p}^{goal} . The task reward is defined as

$$r_t^g = \begin{cases} \exp\left(-3 \frac{\|\dot{\mathbf{p}}_t^{\text{root}} - \mathbf{v}_t^*\|^2}{\|\mathbf{v}_t^*\|^2}\right) & \text{if } \|\mathbf{p}^{\text{goal}} - \mathbf{p}_t^{\text{root}}\| > R, \\ 1 & \text{otherwise,} \end{cases} \quad (11)$$

where $R = 0.5\text{m}$ defines a goal radius around the target lo-

cation, $\dot{\mathbf{p}}_t^{\text{root}}$ denotes the root velocity, and \mathbf{v}_t^* is the desired velocity toward the goal. The goal observation $g_t \in \mathbb{R}^4$ comprises the relative direction to the target, the target velocity, and the distance to the goal. A trial is considered successful if the character reaches within the 0.5 m radius of the target location.

Strike Task. The objective of character is to approach and strike a box. The task reward is defined as

$$r_t^g = 1 - \mathbf{u}^{\text{world}} \cdot \mathbf{u}^{\text{box}}, \quad (12)$$

where $\mathbf{u}^{\text{world}} = [0, 0, 1]$ is the world up vector and \mathbf{u}^{box} is the box’s up vector. The goal observation $g_t \in \mathbb{R}^{15}$ consists of the state of the target box. The success rate is computed based on whether the box tilts beyond 78° , *i.e.*, $\mathbf{u}^{\text{world}} \cdot \mathbf{u}^{\text{box}} < 0.2$.

Heading Task. The heading task requires the character to move in a target direction \mathbf{d} at a specified speed s^* , where the target velocity is defined as $\mathbf{v}^* = s^* \mathbf{d}$. The reward is formulated as

$$r_t^g = 0.7 \exp(-0.25(\|\mathbf{v}^* - \mathbf{v}_d\|^2 + 0.1\|\mathbf{v}_\perp\|^2)) + 0.3 \max(\mathbf{d} \cdot \mathbf{f}, 0), \quad (13)$$

where \mathbf{v}_d is the character’s velocity component along the target direction \mathbf{d} , \mathbf{v}_\perp is the velocity component orthogonal to \mathbf{d} , and \mathbf{f} represents the character’s current heading direction. The goal observation $g_t \in \mathbb{R}^3$ comprises the target direction and the target speed. A trial is considered successful if the velocity error along the target direction, $|\mathbf{v}^* - \mathbf{v}_d|$, is less than 0.5.

C. Reference Motion Dataset

Our base policies were trained using motion capture data from LAFAN1 [7]. For the motion composition task, we used motion clips sourced from the AMASS dataset [16]. The partial tracking task relies on trajectories produced by the generator G , which are layered on top of the base walking motion, which is also from LAFAN1. A summary of all reference motions used across our experiments is provided in Tab. 3.

D. Experiments

D.1. Ablation Study on MI Loss Strength

We conducted an ablation study across a wider range of mask-invariant weights λ_{MI} . The full results are presented in Tab. 4. For each setting, we report the average normalized entropy H_{norm} computed over all body-part masking configurations. To account for different initialization conditions, we evaluate the policies when motions are generated from random initial states (Random Init.) as well as from a same initial state (Same Init.), and we report H_{norm}

Motion	Dataset	Duration (s)	Stage	Task
Jump	LAFAN1	24.4	Base	C
Locomotion	LAFAN1	1000	Base	C
Aim	LAFAN1	305	Base	C
Walk	LAFAN1	5.6	Base	T
Walk (G1)	LAFAN1	33.3	Base	C
Alternating kick	AMASS	9.9	Residual	C
Rotate arms	AMASS	25.1	Residual	C
Sneak	AMASS	8.3	Residual	C
Punch	AMASS	18.3	Residual	C
Punch (G1)	AMASS	14.5	Residual	C
Groom	AMASS	14.5	Residual	C

Table 3. Reference motions used in our experiments. Task type "C" denotes motions used for motion composition, and "T" indicates motions used for partial tracking.

for both cases. Under the random-initialization setting, a moderate-to-strong MI weight yields the highest entropy, with $\lambda_{\text{MI}} = 5.0$ performing best. Beyond this point, excessively large weights begin to harm performance, reducing entropy even below that of the weakest regularization ($\lambda_{\text{MI}} = 0.1$). Under the same-initialization setting, the results reflect the policy’s ability to branch into different motions from an identical initial state. In this case, $\lambda_{\text{MI}} = 1.0$ achieves the highest value, outperforming all weaker and stronger regularization strengths. This suggests that a moderate MI weight provides the best balance between enforcing invariance and preserving sufficient freedom for the policy to diverge into a wide range of skills. In our experiments, we adopt $\lambda_{\text{MI}} = 1.0$ as the default setting.

D.2. Consistency Across Masking Conditions

We further evaluated whether the base policy maintains consistent behavioral diversity under different masking conditions. Specifically, we compute H_{norm} over all masking combinations involving up to two body parts. As shown in Tab. 5, the entropy values vary by only 0.014, indicating that the policy produces similarly diverse motions regardless of which body parts are masked. This consistency confirms that the MI-trained base policy serves as a reliable motion prior for subsequent residual learning.

We also evaluated the consistency across all masking conditions when the MI loss is *not* applied. As shown in Tab. 6, the H_{norm} values decrease substantially across all combinations. The policy retains relatively higher values only when a single body part is masked (up to 0.845), but performance drops further when two body parts are masked. The degradation becomes most severe when the masked parts are critical for locomotion. In particular, masking both legs produces the lowest score of 0.602, showing a large gap compared to the 0.905 achieved when the MI loss is applied in Tab. 5. These results highlight the importance of MI regularization in maintaining consistent behaviors under diverse masking configurations.

Method	Masking	MI Weights (λ_{MI})	Avg. $H_{norm} \uparrow$	
			Random Init.	Same Init.
AMP [19]	×	—	0.9124	0.8890
– Ours w/o \mathcal{L}_{MI}	✓	—	0.7448	0.7019
+ Ours w/ \mathcal{L}_{MI}	✓	0.1	0.8968	0.8387
	✓	0.5	0.8987	0.8441
	✓	<u>1.0</u> (default)	0.9025	0.8833
	✓	2.5	0.9048	0.8387
	✓	5.0	0.9123	0.8552
	✓	10.0	0.9075	0.8722
	✓	50.0	0.8803	0.8461

Table 4. Ablation study on the mask-invariant loss weights.

	Trunk	Left Arm	Right Arm	Left Leg	Right Leg
Trunk	0.909	0.896	0.906	0.908	0.906
Left Arm	—	0.895	0.900	0.898	0.896
Right Arm	—	—	0.907	0.902	0.903
Left Leg	—	—	—	0.904	0.905
Right Leg	—	—	—	—	0.902

Table 5. Normalized entropy (H_{norm}) across all masking conditions. Values are arranged in an upper-triangular matrix, with lower entries omitted (“—”) due to symmetry. Diagonal entries correspond to single body-part masking, while off-diagonal entries represent the average H_{norm} when two body parts are masked simultaneously. Darker shades indicate higher entropy.

	Trunk	Left Arm	Right Arm	Left Leg	Right Leg
Trunk	0.798	0.830	0.790	0.749	0.629
Left Arm	—	0.814	0.736	0.796	0.702
Right Arm	—	—	0.845	0.681	0.664
Left Leg	—	—	—	0.754	0.602
Right Leg	—	—	—	—	0.783

Table 6. Normalized entropy (H_{norm}) across all masking conditions when the MI loss is *not* applied. The H_{norm} values decrease substantially across all combinations, with particularly severe degradation when two body parts are masked simultaneously.

D.3. Frame Visitation Frequency

Fig. 1 shows frame visitation distributions for locomotion under an extreme masking condition where both legs are masked. The y-axis indicates visitation frequency per frame (log scale), and the x-axis lists frames sorted by visitation frequency. Without the MI loss (blue), the distribution is highly skewed, showing that the policy visits only a narrow subset of frames. In contrast, applying the MI loss (yellow) yields a far more uniform distribution, indicating that the policy explores a broad range of motion states rather than collapsing onto a small set of highly visited frames. For reference, we also show the AMP baseline (green), which does not apply masking. Notably, the base policy with the MI loss achieves a level of uniformity comparable to AMP, demonstrating that it supports robust and diverse state visitation even under heavily masked observations.

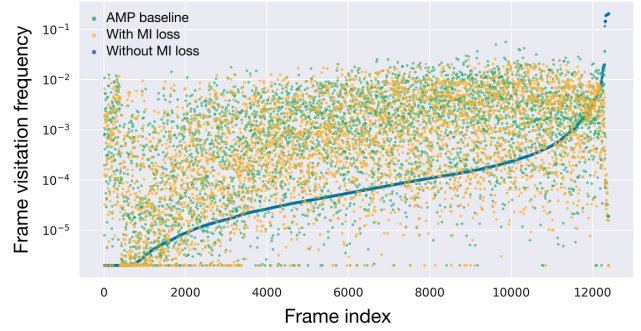


Figure 1. Frame visitation frequencies for locomotion with both legs masked. The policy without MI loss (blue) shows a skewed visitation pattern, whereas applying the MI loss (yellow) produces a markedly more uniform distribution. Our mask-invariant base achieves a level of uniformity comparable to AMP (green), indicating that the MI loss preserves diverse state visitation even under heavy masking.

D.4. Qualitative Comparison for Motion Composition

Fig. 2 visualizes generated motion sequences for the three composition cases using each method. Our method produces more natural and coordinated motion combinations, while CML often exhibits delayed activation or incomplete execution of the adapted motion (e.g., limited arm rotation or missing/delayed kick behaviors). These qualitative results corroborate the entropy-based findings, demonstrating that our framework yields more balanced and expressive motion compositions. Notably, while CML is constrained to fixed adaptation regions, our method supports more complex scenarios and continues to achieve superior performance under these challenging, high-flexibility settings.

D.5. Qualitative Comparison for Partial Motion Tracking

Fig. 3 presents a comparison among our method, CML, and a from-scratch model, all conditioned on the text commands shown on the left. When the character is instructed to exe-

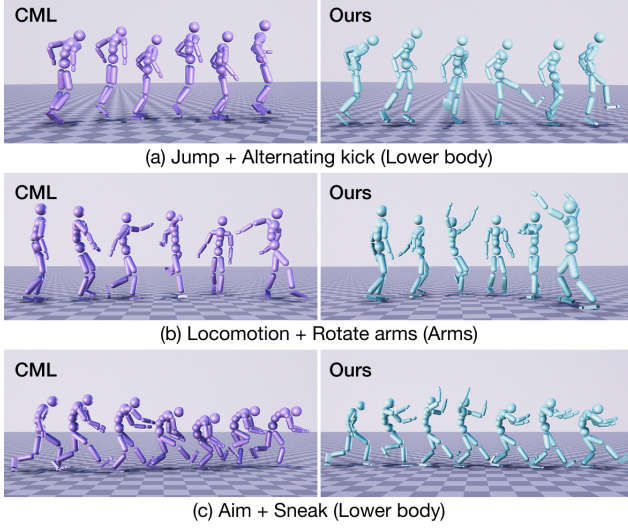


Figure 2. Qualitative comparison of composition methods, each motion generated from the same initial state. In (a), the kick motion in CML is delayed; in (b), CML exhibits severe artifacts and fails to complete full arm rotations; and in (c), CML produces limited motion combinations compared to ours.

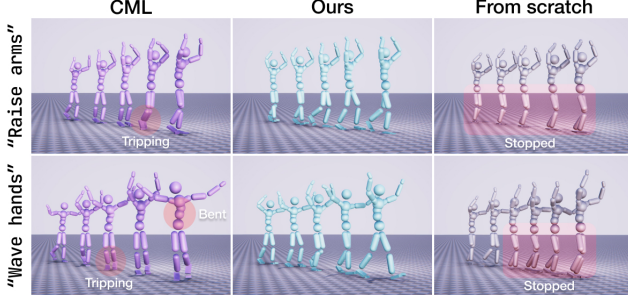


Figure 3. Qualitative comparison of our method, CML, and a from-scratch model for the partial tracking task given the text commands on the left. The red region highlights deviations from the base behavior.

cute more pronounced arm motions, such as raising arms or waving hands, both CML and the from-scratch model often struggle to preserve the underlying walk. CML exhibits intermittent disruptions in its stepping pattern, leading to noticeable gait instability. The from-scratch model performs even less reliably and frequently fails to sustain walking. In contrast, our method maintains stable locomotion throughout the sequence. With its improved stability and stronger tracking capability, our controller produces motions that remain coherently aligned with the intended goals provided by the text commands.

E. Training Procedure

E.1. Training Base Policy

Algorithm 1 summarizes the full training procedure for the base policy. For each rollout, we store the transition in the

Algorithm 1: Training Base Policy

```

Initialize base policy  $\pi$  and value function  $V^\pi$ 
Initialize discriminator  $D^\pi$ 
Initialize replay buffer  $\mathcal{B}$ 

while training not converged do
    /* Collect rollouts */
    for  $t = 1, \dots, L$  do
        Sample mask  $m_t$ 
        Compute masked state  $\bar{s}_t$  from  $(s_t, m_t)$  using Eq. (3)
        Sample action  $a_t \sim \pi(a_t | \bar{s}_t, m_t)$ 
        Simulate next state  $s_{t+1}$  using  $s_t$  and  $a_t$ 
        Store transition  $(s_t, m_t, a_t, s_{t+1})$  in buffer  $\mathcal{B}$ 
         $s_t \leftarrow s_{t+1}$ 
    end

    /* Update discriminator */
    Update  $D^\pi$  on  $(s, s')$  from  $\mathcal{M}$  and  $\mathcal{B}$ 

    /* Update base policy */
    for  $u = 1, \dots, n$  do
        Sample  $(s_t, m_t, a_t, s_{t+1})$  from  $\mathcal{B}$ 
        Compute  $\mathcal{L}_{\text{MI}}$  from  $(s_t, m^0, m_t)$  using Eq. (3)
         $\mathcal{L} \leftarrow \mathcal{L}_{\text{PPO}} + \lambda_{\text{MI}}$ 
        Update  $\pi$  and  $V^\pi$  with  $\mathcal{L}$ 
    end
end

```

replay buffer \mathcal{B} , including the full state s_t and the mask m_t used to construct \bar{s}_t . This is necessary since computing the MI loss requires comparing the action distributions under the masked state \bar{s}_t paired with its mask m^t against those under the corresponding full state s_t with the null mask m^0 .

E.2. Training Residual Policy for Partial Tracking

Algorithm 2 outlines the training procedure for the residual policy in the partial motion tracking task. To improve efficiency, we sampled batches of future trajectories τ_f every N_{sample} iterations using G and store them in a goal buffer \mathcal{B}_τ . For each rollout, a trajectory was sampled from \mathcal{B}_τ , and its future goal states g_t were consumed sequentially to guide the residual policy ψ .

F. Details on Trajectory Generator

The generator G is a diffusion-based autoregressive motion generation model [35] that takes a text prompt c and a history motion $\tau_h = x_{t-H+1:t}$ as conditioning inputs, and predicts a future motion segment $\tau_f = x_{t+1:t+F}$.

Motion Representation. Each pose in the sequence is represented as $x_t = (\mathbf{t}_t, \mathbf{o}_t, \mathbf{p}_t, \Theta_t, \dot{\mathbf{t}}_t, \dot{\mathbf{o}}_t, \dot{\mathbf{p}}_t)$, where $\mathbf{t}_t \in \mathbb{R}^3$ is the root translation, $\mathbf{o}_t \in \mathbb{R}^6$ the root orientation in 6D representation, $\Theta_t \in \mathbb{R}^{(J-1) \times 6}$ the local joint rotations, and $\mathbf{p}_t \in \mathbb{R}^{J \times 3}$ the joint positions. The dotted terms represent the first-order temporal derivatives (velocities) of the corre-

Algorithm 2: Training Residual Policy for Partial Tracking

```

 $\pi$  : pre-trained base policy
 $G$  : pre-trained trajectory generator
 $\mathcal{J}_K$  : pre-defined subset of body parts
 $\mathcal{C}$  : text embedding pool for  $\mathcal{J}_K$ 

Initialize residual policy  $\psi$  and value function  $V^\psi$ 
Initialize discriminator  $D^\psi$ 
while training not converged do
  /* Sample trajectories offline */
  if  $i \bmod N_{\text{sample}} = 0$  then
    Sample random noise  $z_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
    Sample text input  $c \sim \mathcal{C}$ 
    Sample  $\tau_f \sim G(z_T, T, \tau_h, c)$ 
    Store  $\tau_f$  in  $\mathcal{B}_\tau$ 
  end

  /* Collect rollouts */
  Sample  $\tau_f \sim \mathcal{B}_\tau$ 
  for timesteps  $t = 1, \dots, F$  do
    Compute  $\bar{s}_t^K$  from  $(s_t, m_t^K)$  using Eq. (1)
    Compute  $g_t \leftarrow \text{ExtractGoal}(\tau_f; t, K)$ 
    Sample  $a_t \sim \pi(a_t | \bar{s}_t^K, m_t^K)$ 
    Sample  $\Delta a_t \sim \psi(\Delta a_t | s_t, g_t, a_t)$ 
    Compute  $\hat{a}_t = w_{\text{gain}} \cdot a_t + \Delta a_t$ 
    Simulate next state  $s_{t+1}$  using  $s_t$  and  $\hat{a}_t$ 
    Store transition  $(s_t, g_t, m_t^K, \hat{a}_t, s_{t+1})$  in  $\mathcal{B}$ 
     $s_t \leftarrow s_{t+1}$ 
  end

  /* Update discriminator */
  Update  $D^\pi$  on  $(s, s')$  from  $\mathcal{M}$  and  $\mathcal{B}$ 

  /* Update residual policy */
  Update  $\psi$  and  $V^\psi$  using data from  $\mathcal{B}$ 
end

```

sponding quantities. The motion is canonicalized with respect to the first frame of the history motion.

Formulation. The generator G is implemented as a latent diffusion model [5], where generation operates within the compact latent space constructed by the VAE. In the forward diffusion process, Gaussian noise is progressively added to the latent variable z_0 according to

$$q(z_k | z_{k-1}) = \mathcal{N}(\sqrt{\alpha_k} z_{k-1}, (1 - \alpha_k)\mathbf{I}), \quad (14)$$

where $\alpha_k \in (0, 1)$ is a hyperparameter for noise scheduling at diffusion step k . The diffusion model is trained to approximate the reverse process

$$p_\theta(z_{k-1} | z_k, k, \tau_h, c) = \mathcal{N}(\mu_\theta, \tilde{\sigma}_k \mathbf{I}), \quad (15)$$

where μ_θ is modeled by a neural network parameterized by θ , and $\tilde{\sigma}_k$ is the variance specified by the noise schedule. During generation, the model iteratively denoises z_k from $k = T$ down to 0, and the final latent vector \hat{z}_0 is decoded

by the VAE decoder to obtain the clean future motion τ_f . For convenience, we refer to the entire sampling pipeline as $\tau_f = G(z_T, T, \tau_h, c)$.

Implementation details. We used a history length of $H = 5$ and a future length of $F = 15$. The total number of diffusion steps was $T = 10$. Both the VAE and the diffusion model were implemented using Transformer architectures. We performed sampling using the DDPM procedure and applied classifier-free guidance (CFG) with a guidance weight of 5.0 for text-conditioned generation. The text input was encoded using the CLIP [23] text encoder. Following DART [35], we trained the diffusion model using a simple objective, where the model directly predicts the clean latent code rather than the added noise.

G. Details on Text Embedding Pool

In partial tracking task, we defined a subset of body parts \mathcal{J}_K for adaptation and sample a text input c from its associated embedding pool \mathcal{C} as input to G . To construct \mathcal{C} , we used OpenAI’s GPT-4.1 API to automatically determine the body-part involvement of each action label in the BABEL dataset [22], following a similar strategy to [1]. GPT is guided through three sequential steps: (1) Action clarity check: Determine whether the label contains enough information to infer body-part involvement. Vague labels are skipped with a brief justification. (2) Body-part analysis: For analyzable labels, identify the essential body parts based on an isolated recognizability criterion, meaning a part is chosen only if the action remains identifiable when observed in isolation. (3) Structured output: Return a JSON entry containing the selected parts and a brief rationale, or a skip flag with a skip reason. After the body-part assignments are obtained, we extracted only the descriptions associated with the designated parts to construct \mathcal{C} . In our experiments, we constructed \mathcal{C} for the *Arms* region and inputted arms-related texts as conditioning inputs to G to produce the kinematic guidance.

H. Failure Modes

Our method may struggle when conflicting motions are assigned to closely related body parts. For example, combining a *Kick* on one leg with a *Jump* on the other can produce unnatural motion. This is because motions like jumping inherently require coordinated support from multiple related parts, making them difficult to realize when one part is constrained by a different objective.