GaitSpan: Growing Humanoid Locomotion from Walking to Running

Supplementary Materials

Contents

A The Positioning of GaitSpan **2**

B Additional Baseline Comparisons and Analysis **3**
B.1 MuJoCo Comparison and Sim-to-Sim Transfer 3
B.2 What the Seed Walking Policy Encodes 4
B.3 GaitSpan with Different Seed Policies 4
B.4 Unfold Different Transformer Seeds 7

C Additional Ablation Studies **8**
C.1 Looking Inside GaitWave and H-SLIP 8
C.2 The Universality of GaitSpan 9

D Additional Implementation Details **12**
D.1 Evaluation Protocol and Metrics 12
D.2 Additional Method Details 13
D.3 Action Space 15
D.4 General Training Setup 16
D.5 Robot-Specific Configurations 17

Abstract

In this supplementary material, we provide a more detailed elaboration of *GaitSpan*. We first recap the positioning of *GaitSpan* and highlight how it differs from current humanoid locomotion methods (Sec. A). We then present additional baseline comparisons, including the complete Overall / ID / OOD breakdown of the main comparison table, and further analyze the role of the seed walking policy and the robustness of GaitSpan to different seed policy variations (Sec. B). We also provide additional ablation studies, including the complete Overall / ID / OOD breakdown of the main ablation table, that examine the detailed designs of GaitWave and H-SLIP, including wave composition, speed-dependent coefficient usage, hierarchical SLIP shaping, and component generalization across robot embodiments, as well as additional emergent behavior analyses (Sec. C). Finally, we include additional implementation details, including the evaluation protocol and formal metric definitions used for tables and plots (Sec. D).

A The Positioning of GaitSpan

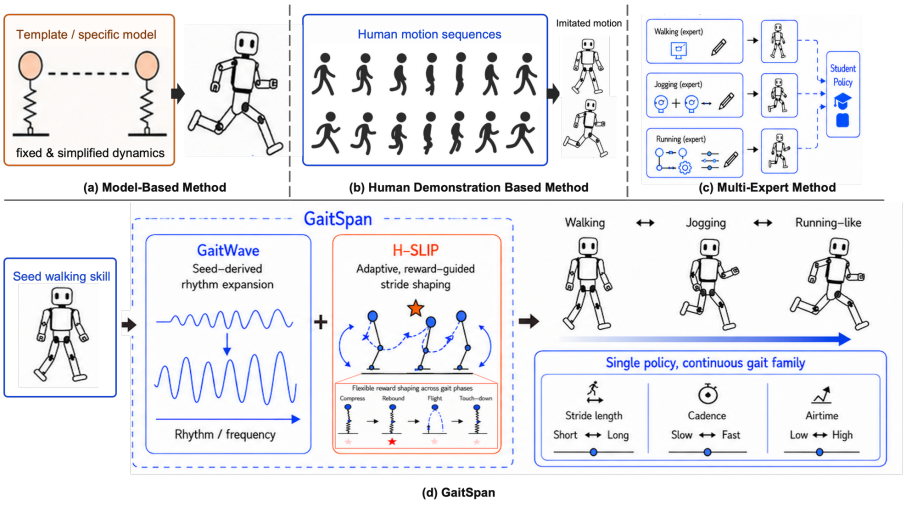


Figure 1: **The conceptual framework differences.** We summarize the key conceptual level differences between our work and current humanoid robot trends on locomotion for better positioning of GaitSpan.

To better understand GaitSpan's positioning, we present a conceptual framework comparison in Fig. 1. Existing approaches to diverse humanoid locomotion typically fall into three paradigms. Model-based methods (Fig. 1 a) construct locomotion through simplified dynamics or specific templates, providing useful physical structure but often limiting flexibility. Human-demonstration-based methods (Fig. 1 b) import rich motion priors without explicitly modeling robot dynamics, but rely on external cross-embodiment data that may mismatch the robot's morphology, actuation, contact dynam-

ics, and energy objectives, while remaining constrained by the demonstrated motion distribution. Multi-expert methods (Fig. 1 c) train separate controllers for skills like walking, jogging, and running, then distill them into one policy. This treats gait diversity as the assembly of independently acquired skills rather than the growth of one skill into another. Such designs can be costly to scale, sensitive to expert boundaries, and prone to discontinuous or conflicting behavior during composition.

In contrast, GaitSpan (Fig. 1d) views walking as a seed skill from which a broader family of related locomotion regimes can grow. Starting from a frozen walking policy, GaitWave generates seed-derived rhythm variations to modulate cadence and timing, while H-SLIP provides reward-guided stride shaping through encouraging representative dynamic events such as compression, rebound, flight, and touchdown. A residual branch captures remaining command- and state-dependent adaptations. The resulting policy spans walking-, jogging-, and running-like regimes with continuous gait changes. This makes GaitSpan a skill-growth framework, where walking is not a terminal behavior or one expert among many, but the seed from which a continuous family of humanoid gaits emerges. By expanding rather than overwriting the seed skill, GaitSpan preserves the reusable structure of walking while allowing new regime-specific dynamics to emerge upon it.

B Additional Baseline Comparisons and Analysis

In this section, we first provide a MuJoCo-aligned qualitative comparison with the generalist controller and examine Isaac Gym→MuJoCo sim-to-sim transfer (Subsec. B.1). We next focus on two questions: **1)** what locomotion structure is already encoded in the seed walking policy, and what benefits and limitations arise from directly modifying its phase (Subsec. B.2)? **2)** How does GaitSpan perform regarding different seed policy variants? (Subsec. B.3)?

B.1 MuJoCo Comparison and Sim-to-Sim Transfer

Setting. The main-paper quantitative comparison evaluates BFM-Zero in its official Isaac Sim setting and GaitSpan in Isaac Gym, keeping each method in its respective native simulation environment for the fairest possible comparison. Here, we additionally provide a qualitative MuJoCo-aligned comparison on Unitree G1 (29,DoF) to examine two questions: **1)**whether the relative behavior remains consistent when both methods are evaluated under the same simulator, and **2)** whether GaitSpan robustly transfers zero-shot from Isaac Gym to MuJoCo.

Results. Fig. 2 shows two findings. 1) *Under MuJoCo, GaitSpan still outperforms the generalist controller.* BFM-Zero fails to maintain stable high-speed locomotion and falls (Fig. 2(a)), whereas GaitSpan produces speed-appropriate gaits without collapse (Fig. 2(c)). 2) *Sim-to-sim transfer is strong.* The Isaac Gym and MuJoCo rollouts of the same GaitSpan policy (Fig. 2(b)–(c)) remain consistent across commanded speeds, supporting zero-shot Isaac Gym→MuJoCo transfer beyond the training simulator used for table metrics.

a) Generalist Controller (BFM-Zero), MuJoCo



b) GaitSpan (Ours), Isaac Gym



c) GaitSpan (Ours), MuJoCo

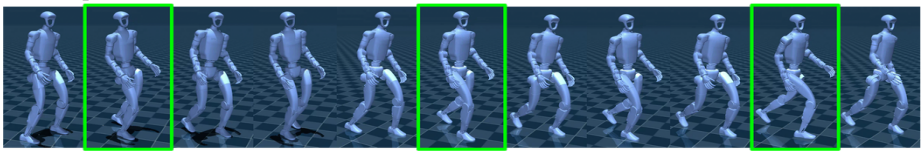


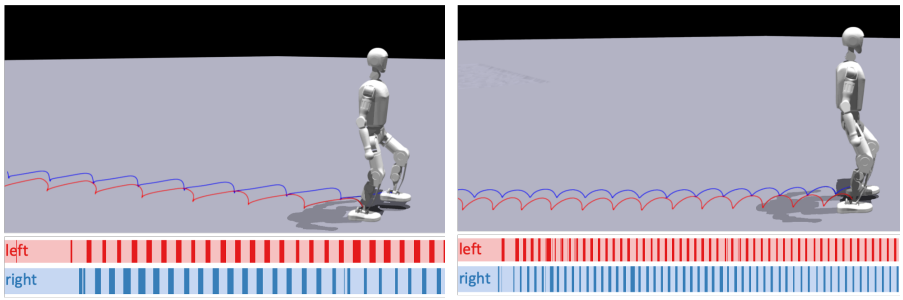
Figure 2: **MuJoCo-aligned qualitative comparison and sim-to-sim transfer on Unitree G1 (29 DoF).** From left to right, commanded speeds increase from 0.5 to 1.5 and 2.5 m/s. **(a)** BFM-Zero evaluated in MuJoCo becomes unstable and falls at high speed. **(b)–(c)** GaitSpan remains speed-adaptive and stable in Isaac Gym and after zero-shot transfer to MuJoCo, indicating robust sim-to-sim transfer.

B.2 What the Seed Walking Policy Encodes

We first unfold the frozen seed walking policy to understand what locomotion structure it already contains. As shown in Fig. 3, changing the phase frequency of the seed policy alters the rhythm and contact density of the resulting walking motion, indicating that the seed policy encodes reusable rhythmic structure. However, directly changing the phase frequency is not sufficient for command-adaptive gait growth. Associating with Fig. 5 (a) in the main paper, when the target forward velocity increases beyond the seed policy's training range, such as $v_x = 1.5$ m/s compared with the training range of $[-1, 1]$ m/s, the phase-scaled seed still produces similar rhythmic and contact patterns but leads to large tracking error. These results suggest that the seed policy contains useful rhythmic primitives, but they must be used in a way that is adaptive rather than directly executed. This motivates the design philosophy of GaitWave, where the seed policy should serve as a stabilizing anchor that preserves its reusable rhythmic structure, while its seed-derived action waves can be adaptively recomposed in a certain way to move beyond the seed's original command range.

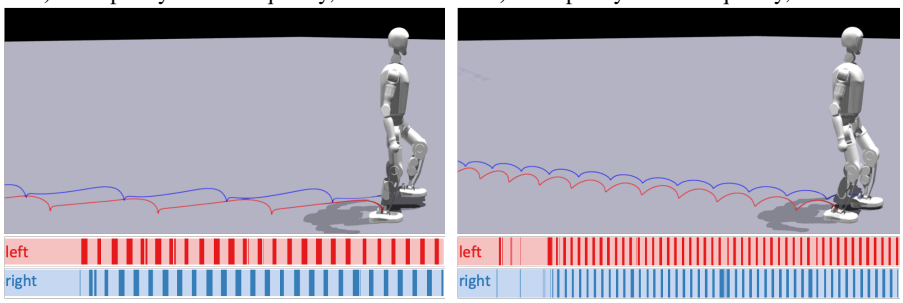
B.3 GaitSpan with Different Seed Policies

To test whether GaitSpan depends on a particular seed-policy implementation, we replace the original seed with policies trained under substantially different configurations.



a) Seed policy at 1.0 frequency, 0.5m/s

b) Seed policy at 2.0 frequency, 0.5m/s



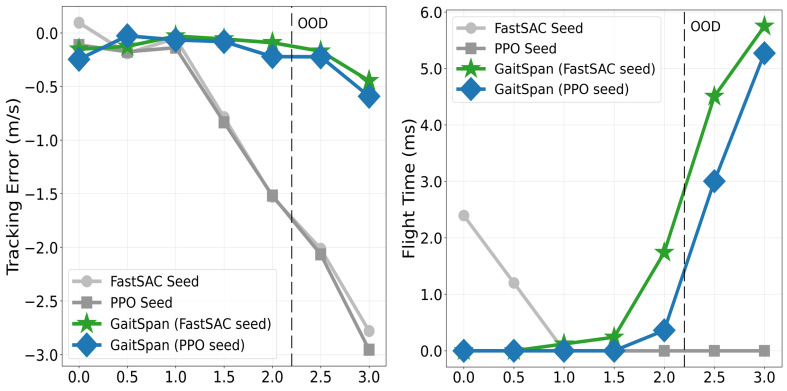
c) Seed policy at 1.0 frequency, 1.5m/s

d) Seed policy at 2.0 frequency, 1.5m/s

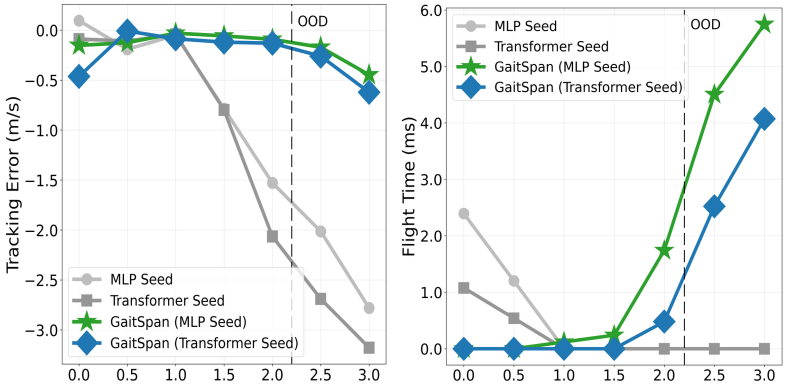
Figure 3: **Phase scaling of the seed policy.** We vary the phase frequency and velocity command of the frozen seed policy on the Booster T1-23DoF robot. Foot trajectories and foot-contact plots are shown at 0.5 m/s and 1.5 m/s. From the trajectories and plots, we can see that phase scaling changes the contact rhythm and foot motion. This indicates that the seed walking policy contains reusable rhythmic structures. However, direct phase scaling reuses this structure only in a fixed manner and does not produce reliable high-speed gait growth beyond the seed policy's training range.

Settings. We freeze each alternate walking seed and grow residual skill with the same GaitSpan recipe (GaitWave + H-SLIP; residual trained with FastSAC), using the same continuous v_x ramp protocol as elsewhere in this supplementary material (the overall metric covers both ID and OOD, where ID: $v_x^{\mathrm{cmd}} \leq 2.2$ m/s, OOD: $v_x^{\mathrm{cmd}} > 2.2$ m/s). **1)** *RL algorithm.* We compare FastSAC and PPO walking seeds, and the corresponding GaitSpan policies grown from each. **2)** *Actor architecture.* We compare an MLP seed with a Transformer seed, and GaitSpan grown from each. Here "Transformer Seed" denotes the grouped-token Transformer (one token per observation term).

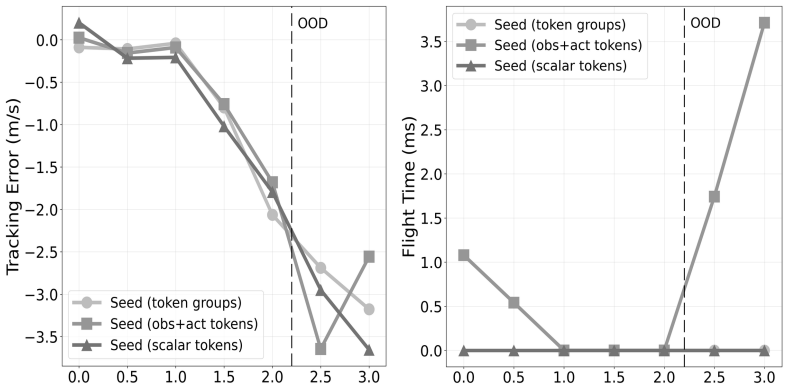
Results. Fig. 4 and Tab. 1–2 show several key findings. 1) *GaitSpan is not tied to FastSAC.* Replacing the FastSAC MLP seed with a PPO seed yields a similarly limited frozen walking policy (Overall MTE 1.00/1.04). Growing GaitSpan from either seed restores high-speed tracking (Overall MTE 0.12/0.14, Overall TSR 98%/97%) and enables controlled flight under faster commands (Fig. 4(a), Tab. 1). 2) *GaitSpan is not tied to an MLP actor.* The Transformer (grouped-token) seed alone remains limited in tracking ability across the speed range, but GaitSpan grown from it follows the same trend as the MLP-seed GaitSpan run (Overall MTE 0.15, Overall TSR 98%),



a) FastSAC vs. PPO seed



b) MLP vs. Transformer seed



c) Transformer tokenization (seeds only)

Figure 4: **GaitSpan with different seed policies on Booster T1 (23 DoF).** Left: absolute tracking error; right: flight time. **(a)** Growing from FastSAC or PPO walking seeds. **(b)** Growing from MLP or Transformer (grouped-token) seeds. **(c)** Frozen Transformer seeds under different tokenization schemes (no GaitSpan growth). Vertical dashed lines mark the ID/OOD boundary at 2.2 m/s.

Table 1: FastSAC vs. PPO seed policies (Booster T1, 23 DoF). Best entries are shown in **bold**.

Method	**MTE** $\downarrow$	**TSR (%)** $\uparrow$	**CoT** $\downarrow$
FastSAC Seed	1.00	17.7	0.71
PPO Seed	1.04	15.1	**0.62**
GaitSpan (FastSAC seed)	**0.12**	**98.3**	0.81
GaitSpan (PPO seed)	0.14	97.0	0.93

Table 2: MLP vs. Transformer (grouped-token) seed policies (Booster T1, 23 DoF). Best entries are shown in **bold**.

Method	**MTE** $\downarrow$	**TSR (%)** $\uparrow$	**CoT** $\downarrow$
MLP Seed	1.00	17.7	0.71
Transformer Seed	1.22	22.3	**0.60**
GaitSpan (MLP Seed)	**0.12**	**98.3**	0.81
GaitSpan (Transformer Seed)	0.15	97.9	0.88

maintaining low tracking error while developing flight at high speed (Fig. 4(b), Tab. 2). Overall, despite these changes to the seed policy, GaitSpan consistently expands the resulting walking policies into faster and more dynamic gait regimes, while maintaining stable command tracking. These results suggest that the proposed framework is not tied to a specific RL algorithm or network architecture, but generalizes across different seed policies. This robustness stems from GaitSpan not blindly inheriting the seed actions; instead, it treats them as structured motor priors that can be transformed and adaptively corrected during skill growth.

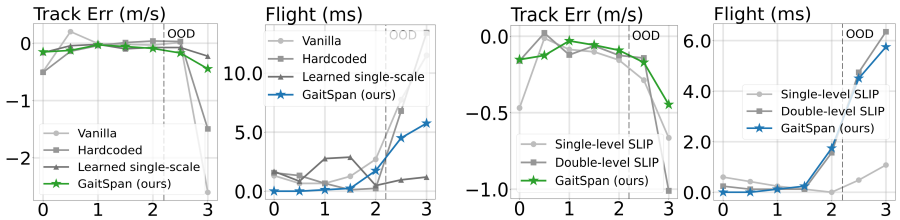
B.4 Unfold Different Transformer Seeds

As Transformer policies can represent the same walking skill with different internal structures, we further unfold the seed design itself to study how tokenization affects its locomotion performance and extrapolation.

Specifically, we ablate three frozen Transformer seeds: *token groups* (one token per observation term), *obs+act tokens* (one observation token and one action token), and *scalar tokens* (one token per scalar dimension of $[\mathrm{obs}\|a_0]$). Among frozen Transformer seeds, grouped tokens achieve the best Overall/ID tracking success, while all three still fail under OOD commands without GaitSpan (OOD TSR 0%; Fig. 4(c), Tab. 3). Under the surface, scalar tokenization is computationally inefficient and slow (as the sequence length scales with the flat observation/action dimension (on the order of 10^2 tokens), and treats each scalar as an independent token, so it does not reflect the natural interrelationships within observation groups. Conversely, packing the entire observation into a single token (obs+act) is overly simplistic: it collapses heterogeneous sensory structure into one embedding and leaves little room for the Transformer to reason about relationships among observation terms. Grouped tokenization strikes a practical middle

Table 3: Transformer tokenization ablation on frozen seeds **only** (Booster T1, 23 DoF). Best Overall / ID / OOD entries are shown in **bold**. "–" means CoT is undefined ($|v_x| < 0.2$ m/s).

	MTE ↓			TSR (%) ↑			CoT ↓		
Method	Overall	ID	OOD	Overall	ID	OOD	Overall	ID	OOD
Seed (token groups)	**1.22**	0.64	**2.81**	**22.3**	**30.4**	0.0	**0.60**	**0.56**	1.00
Seed (obs+act tokens)	1.37	**0.60**	3.47	15.6	21.3	0.0	0.71	0.63	0.84
Seed (scalar tokens)	1.31	0.72	2.95	13.1	17.9	0.0	0.70	0.70	**0.69**



a) different combination of wave basis b) SLIP reward

Figure 5: **Ablation on details of GaitWave and H-SLIP designs on Booster T1 robot (23 DoF). (a)** Hierarchical learned wave composition improves tracking and produces controlled flight compared with the variants. **(b)** H-SLIP better balances tracking accuracy and dynamic flight with three-level virtual legs.

ground, preserving within-group structure while keeping the sequence short enough to train efficiently; we therefore use it as the default Transformer seed in Subsec. B.3 setting **2)**.

C Additional Ablation Studies

In this section, we first provide additional ablation studies to better understand the detailed designs of GaitWave and H-SLIP that connect learned rhythm composition with physically guided stride shaping (Sec. C.1). Then, we further examine the universality of GaitSpan across embodiments (Sec. C.2) beyond the results covered in the main paper.

C.1 Looking Inside GaitWave and H-SLIP

In this subsection, we provide additional evidence for how GaitSpan grows a walking skill into richer locomotion regimes.

C.1.1 GaitWave: From Seed Actions to Wave Composition

Ablation on how seed action waves combine. We compare with two variants: a vanilla, *i.e.,* a hardcoded phase-scaling variant that changes with speed, and a learned single-scale variant with 16 bins. As shown in Fig. 5 (a), directly using the seed policy or manually scaling its phase is insufficient for reliable high-speed gait growth, as these

variants either degrade tracking under OOD commands or produce poorly controlled flight. Learning a single-phase scale improves flexibility, but still lacks the command-dependent expressiveness (low flight time in the figure) needed to span the full speed range. In contrast, GaitSpan maintains low tracking error while producing increasing, controlled flight time at high speeds, showing that hierarchical learned composition of seed-derived action waves is essential for growing the walking seed into dynamic gait regimes. We provide additional qualitative comparison to other alternatives in Fig. 6 to reveal further insight. Under the surface, when one shared predictor is used over the entire command domain, learning is dominated by the larger action deviations required at high speeds, causing overly aggressive wave mixtures to leak into the walking regime. Dividing the command space into independent regions reduces this global bias, but removes continuity between neighboring regions and introduces visible boundary artifacts when combination is rigidly manual scaling. Our hierarchical multi-resolution memory resolves this trade-off by sharing coarse trends across speeds while retaining local command-dependent refinements, enabling distinct gait regimes to emerge without sacrificing smooth transitions.

C.1.2 H-SLIP: Why We Need Hierarchical SLIP Reward

We further ablate the hierarchy design in H-SLIP by comparing single-level, double-level, and the final three-level virtual-leg shaping. As shown in Fig. 5(b), single-level SLIP shaping provides only a coarse root–foot dynamic prior. It can encourage limited flight, but ignores the articulated motion of the leg and therefore cannot sufficiently coordinate local compression and rebound across the limb, leading to weaker dynamic gait emergence. Double-level SLIP increases flight emergence by adding local structure to knee and foot, but does not explicitly constrain these local motions to form a coherent whole-body stride. Consequently, the two segments may satisfy their own objectives while producing poorly coordinated root motion, leading to degraded tracking under high-speed and OOD commands.

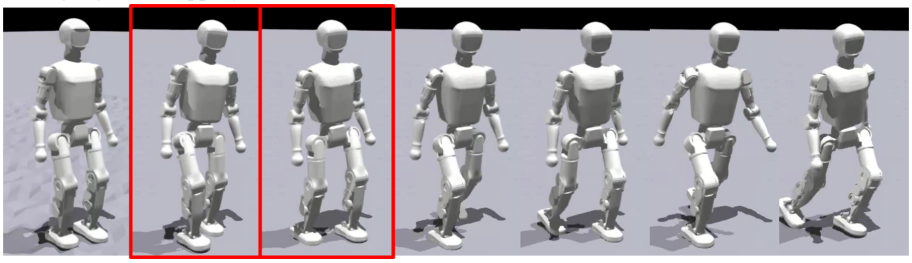
In contrast, the three-level H-SLIP better balances global and local dynamics. The root–foot level captures whole-body support and aerial transition, while the upper- and lower-leg levels shape limb-level compression and rebound. This hierarchy allows the policy to develop flight-producing behaviors without sacrificing tracking accuracy. These results show that H-SLIP is not merely adding a running-style reward; its hierarchical structure is important for turning rhythm-expanded motions into physically meaningful dynamic strides.

C.2 The Universality of GaitSpan

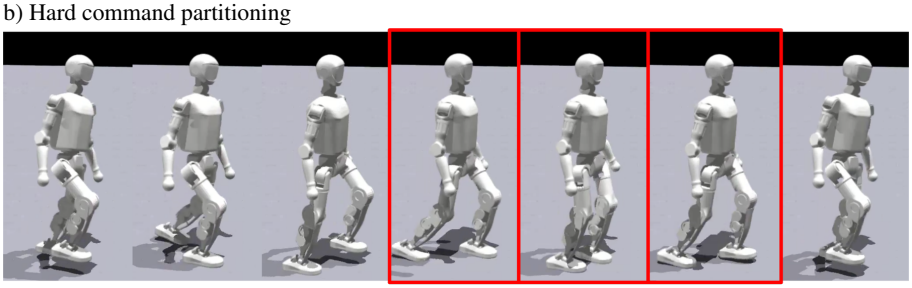
C.2.1 Main Component Generalization Across Robots

We evaluate whether the main components of GaitSpan remain effective across robot embodiments. The experiment repeats the component, training-strategy, wave-composition, and H-SLIP hierarchy ablations on an additional humanoid embodiment, Unitree G1 robot (23 DoF).

a) Single global mapping



b) Hard command partitioning



c) GaitSpan (Ours)

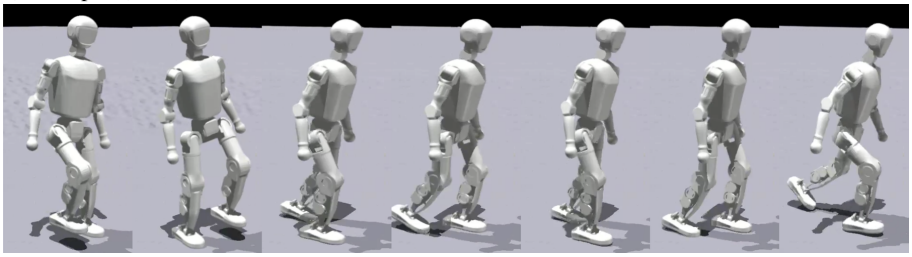


Figure 6: **Qualitative ablation of hierarchical multi-resolution coefficient memory.** a) Using a single global mapping from command speed to coefficients leads to a bias toward high-velocity commands, ignoring walking gaits at lower speeds because the single mapping is too coarse to cover the broad command range, as marked by red boxes. b) Partitioning the command range introduces hard transitions at partition boundaries, causing abrupt gait switches, as marked by red boxes.

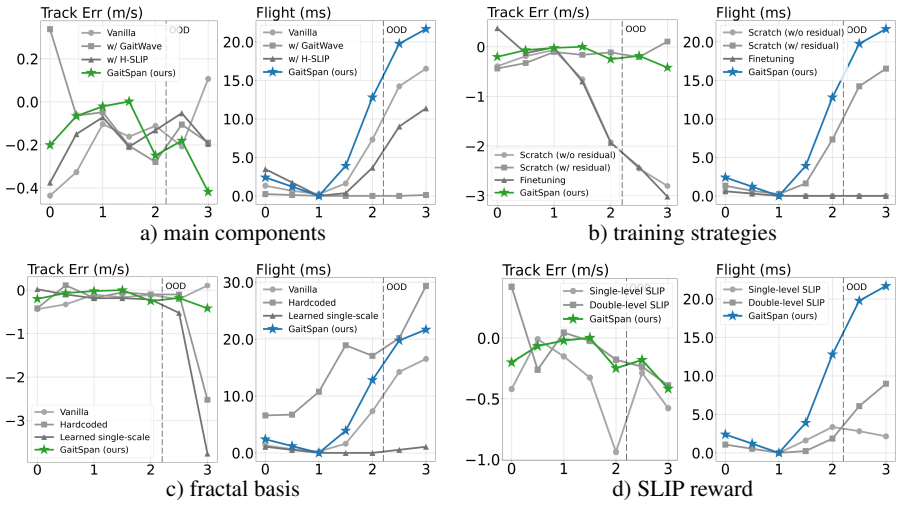


Figure 7: **Additional ablation on Unitree G1 (23 DoF).** The trends are consistent with Booster T1 in the main paper and Fig. 5 in the S.M..

Fig. 7 reveals several key findings. **1)** The overall trends are consistent with those observed on Booster T1: GaitWave and H-SLIP remain complementary, and the training strategy continues to matter. The full GaitSpan model achieves the best balance between tracking accuracy and dynamic flight emergence, while single-component variants, from-scratch training, and direct fine-tuning fail to achieve both simultaneously. **2)** Compared with Booster T1, Unitree G1 shows more significant sensitivity to dynamic shaping, making the benefit of hierarchical H-SLIP more visible. In particular, the gap between single-/double-level SLIP shaping and the full three-level H-SLIP is larger, suggesting that articulated virtual-leg hierarchy is especially important for coordinating global support and local limb dynamics on this embodiment. Together, these results indicate that GaitSpan's design generalizes across robots.

C.2.2 Additional Emergent Behavior Analysis Across Robots

Foot trajectories across speed regimes. Fig. 8 visualizes left and right foot trajectories across five humanoid embodiments as commanded speed increases. **1)** Across robots, the trajectories all evolve from short, low-clearance walking steps to longer and more dynamic patterns in striding, jogging, and running-like regimes. This progression indicates that GaitSpan does not merely speed up a fixed walking gait; instead, it reorganizes stride length, foot clearance, and swing geometry as the locomotion regime changes. **2)** The exact trajectory shapes differ across embodiments, reflecting the flexibility of GaitSpan that enables morphology-specific gait expression under the same skill-growth framework. For example, Booster K1 is a smaller humanoid with $95cm$ height, so its emerged gait appears more compact, whereas larger embodiments such as Booster T1 and Unitree G1 exhibit more significant stride extension and swing variation at higher speeds.

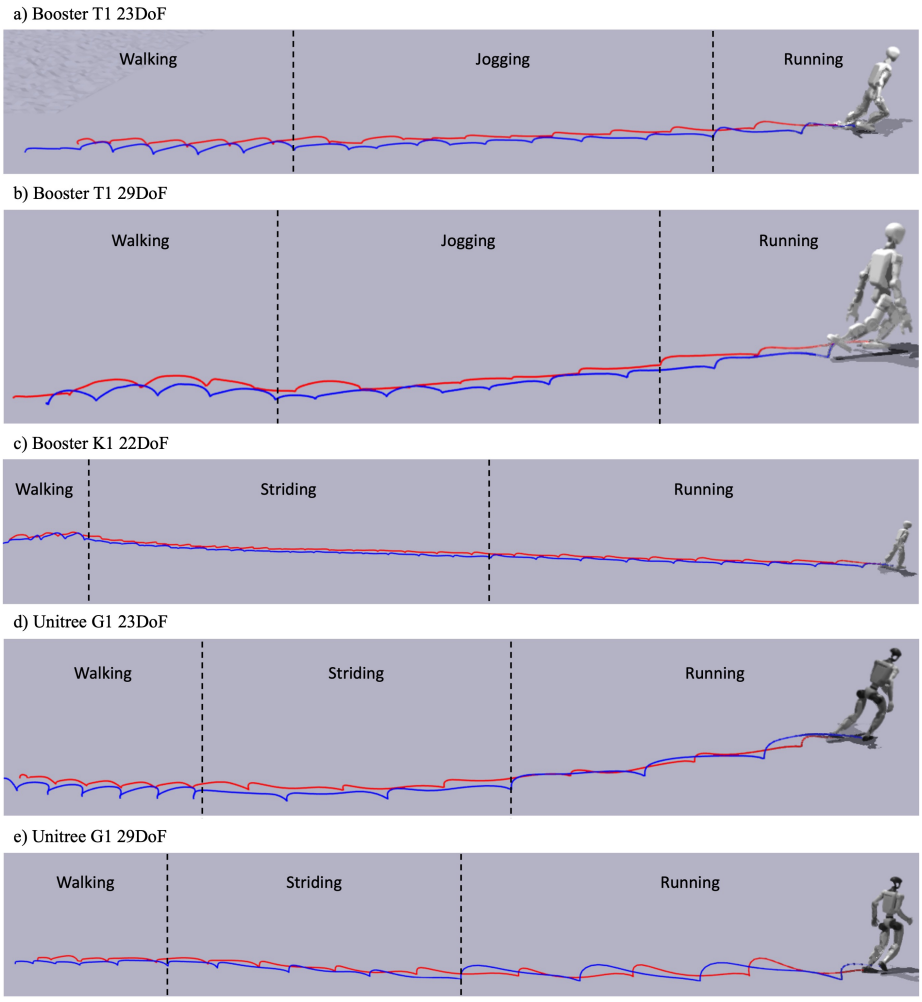


Figure 8: **Foot trajectories across speed regimes and embodiments.** The command speed increases progressively from 0 to 2.5 m/s.

D Additional Implementation Details

D.1 Evaluation Protocol and Metrics

Simulators. For fair comparison, we evaluate each method in the simulator it was trained on. GaitSpan and AMP are trained and evaluated in Isaac Gym. BFM-Zero is trained and quantitatively evaluated in Isaac Sim. Qualitative BFM-Zero videos can appear MuJoCo-like only because the official codebase renders Isaac Sim state with a MuJoCo visualizer (`IsaacRendererWithMuJoco`); the underlying dynamics used for metrics remain Isaac Sim. Separately, we also demonstrate zero-shot sim-to-sim

transfer of GaitSpan from Isaac Gym to MuJoCo. This transfer setting is not used for the AMP/BFM-Zero comparison metrics.

Command schedule. Unless otherwise noted, quantitative evaluations for GaitSpan, AMP, and other continuous controllers in the main paper and this supplementary material—including temporal curves (tracking error, flight time, and energy) and aggregate table metrics—use the same continuous forward-velocity command ramp. The commanded speed v_x^{cmd} increases linearly from 0 to $3.0\,\mathrm{m/s}$ over 1000 control steps at $50\,\mathrm{Hz}$ ($\Delta t = 0.02\,\mathrm{s}$), while lateral and yaw commands are held at zero. We treat commands with $v_x^{\mathrm{cmd}} \leq 2.2\,\mathrm{m/s}$ as in-distribution (within the high-speed training exposure) and $v_x^{\mathrm{cmd}} > 2.2\,\mathrm{m/s}$ as out-of-distribution.

BFM-Zero is evaluated with a different command schedule because it is conditioned by reward-inferred latent prompts rather than a continuously varying velocity command. Following its official reward-prompting interface on Isaac Sim, we use a discrete plateau protocol: for each labeled forward speed $v \in \{0.0, 0.1, \ldots, 3.0\}\,\mathrm{m/s}$ (corresponding to a reward-inferred latent $z(v)$), we condition the policy on $z(v)$ and hold that prompt for 500 control steps, discarding the first 150 warmup steps and recording the remaining 350 steps against the fixed command $v_x^{\mathrm{cmd}} = v$. Each plateau is reset to the default pose before evaluation; if the episode terminates mid-plateau, we reset and re-warm so that recovery frames are not logged. The resulting traces therefore cover 31 constant-speed plateaus over $[0, 3.0]\,\mathrm{m/s}$ (no continuous ramp). We apply the same ID/OOD split as above ($v_x^{\mathrm{cmd}} \leq 2.2$ vs. $> 2.2\,\mathrm{m/s}$), and compute aggregate metrics over all recorded plateau steps with the same definitions below.

Aggregate metrics. From the recorded traces above we report three scalars that summarize overall tracking accuracy, robust command following, and locomotion efficiency.

Mean tracking error (MTE). MTE is the mean absolute forward-velocity error $|v_x - v_x^{\mathrm{cmd}}|$ over all recorded evaluation steps.

Tracking success rate (TSR). TSR is the percentage of recorded steps whose absolute tracking error stays within a hybrid threshold $\tau = \max(0.1,\ 0.20\,|v_x^{\mathrm{cmd}}|)\,\mathrm{m/s}$.

Cost of transport (CoT). CoT measures locomotion efficiency as

$$\mathrm{CoT} = \frac{\sum_t E_t}{mg \sum_t |v_x|\,\Delta t},$$

where $E_t = \left(\sum_j |\tau_{j,t}\,\dot{q}_{j,t}|\right)\Delta t$ is the mechanical joint energy at control step t, m is the robot mass, g is gravitational acceleration, and v_x is the actual base forward velocity. Steps with $|v_x| < 0.2\,\mathrm{m/s}$ are excluded.

D.2 Additional Method Details

D.2.1 Self-Anchored (SA) Regularization

Although GaitWave and the residual branch provide expressiveness, unconstrained expansion can cause the learned controller to disregard the foundational walking skill. We introduce self-anchored regularization to ensure that deviations from walking are deliberate and command-driven.

Seed anchoring. Let g denote a walking-regime gate. At low commanded speeds, we encourage the final action to remain close to the nominal canonical walking action:

$$\mathcal{L}_{\text{anchor},t} = g_t \left\| \mathbf{a}_t - \mathbf{a}_t^{\text{seed}} \right\|_2^2 . \tag{1}$$

This regularization preserves canonical stability in the region where walking is sufficient, while smoothly relaxing the constraint as dynamic motion becomes necessary.

Residual compactness. We discourage unnecessarily large residual corrections through

$$\mathcal{L}_{\text{res},t} = \left\| \mathbf{a}_t^{\text{res}} \right\|_2^2 . \tag{2}$$

This term biases the learned controller toward explaining motion through the expanded canonical structure whenever possible, and using the residual branch only for adaptations not captured by GaitWave.

Phase-Aware self-similarity. To preserve coherent locomotion structure across repeated phase states, we extract a compact proprioceptive feature

$$\mathbf{f}_t = \varphi(\mathbf{s}_t), \tag{3}$$

where $\varphi(\cdot)$ includes joint configurations, joint velocities, gravity-aligned orientation, and optionally contact features. We discretize the locomotion phase into N_ϕ bins and maintain an exponential-moving-average prototype $\bar{\mathbf{f}}_j$ for each phase bin j. For $j_t = \text{bin}(\phi_t)$, the prototype is updated as

$$\bar{\mathbf{f}}_{j_t} \leftarrow (1-\mu)\bar{\mathbf{f}}_{j_t} + \mu \mathbf{f}_t, \tag{4}$$

without gradient propagation through the prototype. The self-similarity loss is

$$\mathcal{L}_{\text{ss},t} = \left\| \mathbf{f}_t - \bar{\mathbf{f}}_{j_t} \right\|_2^2 . \tag{5}$$

This term encourages recurrent locomotion structure without enforcing a predefined gait trajectory or labeled phase schedule.

Temporal action consistency. Finally, we encourage temporal smoothness relative to the canonical action evolution. Let

$$\text{sim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\|_2 \|\mathbf{y}\|_2 + \epsilon}. \tag{6}$$

We define

$$\begin{aligned} \mathcal{L}_{\text{temp},t} = \text{ReLU}\Big(& \text{sim}\left(\mathbf{a}_t^{\text{seed}}, \mathbf{a}_{t-1}^{\text{seed}}\right) \\ & - \text{sim}\left(\mathbf{a}_t, \mathbf{a}_{t-1}\right) \Big). \end{aligned} \tag{7}$$

This term prevents the expanded policy from becoming less temporally coherent than the foundational skill from which it grows.

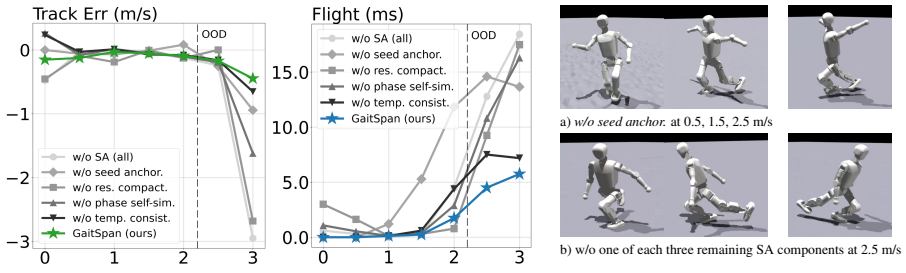


Figure 9: **Ablation on self-anchored (SA) regularization.** Removing self-anchored terms preserves reasonable leg-level gait structure due to GaitWave and H-SLIP, but leads to unstable upper-body motion and less controlled high-speed behavior. The full model better balances tracking, flight emergence, and whole-body stability.

D.2.2 Ablation on SA

We further ablate the auxiliary self-anchored regularization terms used in GaitSpan. As shown in Fig. 9, removing all self-anchored regularization still produces reasonable leg motion, suggesting that the main GaitWave and H-SLIP designs already provide the core structure for gait emergence. However, the resulting motions show unstable upper-body behavior and exaggerated posture changes, which may increase risk during real-world deployment. This indicates that self-anchored regularization is not the primary source of gait growth, but serves as an important stabilizer for preserving whole-body coherence.

The individual ablations further show that different regularizers play complementary roles. Removing seed anchoring causes the policy to deviate more aggressively from the walking prior, leading to larger flight time but less controlled motion. Removing residual compactness, phase self-similarity, or temporal consistency also degrades the balance between tracking and dynamic gait emergence, especially under high-speed and OOD commands. In contrast, the full GaitSpan model maintains low tracking error while producing moderate, command-dependent flight, showing that self-anchored regularization helps the policy expand the seed skill without overwriting its stabilizing structure.

D.3 Action Space

At each time step, the policy receives a concatenated observation vector s_t that combines proprioception, velocity command, and short-term history: 1) base angular velocity $\omega_t \in \mathbb{R}^3$, 2) gravity direction projected to the body frame $g_t \in \mathbb{R}^3$ (a compact representation of base orientation), 3) commanded velocities, 4) joint positions $q_t \in \mathbb{R}^{n_{\text{dof}}}$, 5) joint velocities $\dot{q}_t \in \mathbb{R}^{n_{\text{dof}}}$, 6) the previous action $a_{t-1} \in \mathbb{R}^{n_{\text{dof}}}$ (we use 1 frame in practice), and 7) a periodic phase encoding $(\sin\varphi_t, \cos\varphi_t)$ to facilitate learning of cyclic locomotion structure and to support the rhythm expansion described in the main paper. Following common practice in humanoid RL, these target angles are converted into joint torques using a joint-space PD controller with zero target joint velocity:

$$\tau_t = K_p(\hat{q}_t - q_t) - K_d\dot{q}_t, \tag{8}$$

where K_p and K_d are joint-wise gains.

D.4 General Training Setup

We train with FastSAC in Isaac Gym (200 Hz physics, 50 Hz control) using 4096 parallel environments, 20 s episodes, 10 s command resampling, and a 0.2 standing-command probability. We partition the command speed range into three parallel resolutions (2, 4, and 8 bins) and use temperature-scaled softmax blending ($\tau = 0.08$). An anchor regularizer keeps learned weights near $\boldsymbol{\beta} = (1.0, 0.25, 0.10, 0.05)$ over scales $(1.0, 1.25, 1.5, 2.0)$, with the stronger effect at lower velocity. High-speed exposure increases via a sampling bias whose exponent ramps linearly from 1.0 to 2.6 over the first 95% of training, concentrating mass near 2.2 m/s. Learning rates are 3×10^{-4} (actor, critic, entropy), with buffer size 1024, batch size 8192, 8 updates per step, policy updates every 4 steps, $\gamma = 0.97$, Polyak $\tau = 0.125$, a 101-atom distributional critic on $[-20, 20]$, hidden sizes 512/768, symmetry augmentation, automatic entropy tuning, and bf16 training.

Complete **RL reward terms** and **domain randomizations** are provided in Tab. 4 and Tab. 5.

Table 4: **General reward terms and weights.** Per-step contribution is $r_t = \sum_i w_i \, \Delta t \, r_i(\cdot)$.

Reward	**Definition**	**Weight**
Command Tracking		
Tracking lin. vel.	$\exp(-\lVert \mathbf{v}_{xy}^{\text{cmd}} - \mathbf{v}_{xy} \rVert^2 / \sigma)$	3.0
Tracking ang. vel.	$\exp(-(\omega_z^{\text{cmd}} - \omega_z)^2 / \sigma)$	0.15
Gait & Running		
Feet phase	$\exp(-\sum_{f \in \{L,R\}} (z_f - \hat{z}_f(\phi_f))^2 / \sigma)$	1.2
Flight time	$\text{clamp}(t_f / t_{\max}, 0, 1)$	3.6
Global SLIP	defined in Sec. 3	4.8
Local SLIP-Up	defined in Sec. 3	3.6
Local SLIP-Low	defined in Sec. 3	3.6
Posture & Stability		
Penalty ang. vel. (pitch)	ω_y^2	-0.3
Penalty orientation	$\lVert \mathbf{g}_{xy}^{\text{proj}} \rVert^2$	-0.1
Penalty base height	$(h_{\text{base}} - h^*)^2$	-1.0
Penalty original pose	$\sum_j w_j^{\text{dof}} (q_j - q_j^{\text{def}})^2$	-0.05
Regularization		
Penalty action rate	$\lVert \mathbf{a}_t - \mathbf{a}_{t-1} \rVert^2$	-0.2
Penalty close feet	$\mathbb{1}[d_\perp < d^*]$	-1.0
Penalty feet orientation	$\sum_{f \in \{L,R\}} \lVert \mathbf{g}_{xy,f} \rVert$	-0.5
Penalty ankle roll action	$\sum_{f \in \{L,R\}} \lvert a_{ankle_roll,f} \rvert$	-0.1
Survival		
Alive	1	0.2

Table 5: **Domain randomization terms** used for all settings and embodiments.

Randomization	**Range**
Push Disturbances	
Push interval (s)	$[12,\ 20]$
Push $\lvert v_{xy}\rvert$ cap (m/s)	0.22
Control & Actuation	
Action delay (steps)	$[0,\ 1]$
DOF default bias (rad)	$[-0.004,\ 0.004]$
k_P / k_D scale	$[0.9,\ 1.1]$
Torque RFI	off
Physical Properties	
Link mass scale	$[0.93,\ 1.07]$
Torso mass add (kg)	$[-0.35,\ 1.5]$
Friction (abs)	$[0.65,\ 1.1]$
Base CoM offset (m, xyz)	± 0.008
Reset Initialization	
Reset joint pos scale	$[0.94,\ 1.06]$
Reset joint vel (rad/s)	$[-0.08,\ 0.08]$

D.5 Robot-Specific Configurations

We provide robot-specific configurations in Tab. 6 and pose reward weights for every joint in Tab. 7.

Table 6: **Robot-specific configurations.**

Term	**Booster T1**	**Booster K1**	**Unitree G1**
Desired base height h^*	0.68 m	0.58 m	0.75 m
Close feet threshold d^*	0.15 m	0.15 m	0.20 m
Training iterations	100K	100K	50K

Table 7: **Per-joint pose penalty weights w_j^{dof}.** "–": joint not actuated.

Joint	T1-23DoF	T1-29DoF	K1-22DoF	G1-23DoF	G1-29DoF
Head					
Head yaw	50.0	50.0	50.0	–	–
Head pitch	50.0	50.0	50.0	–	–
Left arm					
Shoulder pitch	0.04	0.04	50.0	50.0	50.0
Shoulder roll	0.12	0.12	50.0	50.0	50.0
Shoulder yaw	–	–	–	50.0	50.0
Elbow pitch	0.06	0.06	50.0	–	–
Elbow yaw	0.22	0.22	50.0	–	–
Elbow	–	–	–	50.0	50.0
Wrist pitch	–	50.0	–	–	50.0
Wrist yaw	–	50.0	–	–	50.0
Hand/wrist roll	–	50.0	–	50.0	50.0
Right arm					
Shoulder pitch	0.04	0.04	50.0	50.0	50.0
Shoulder roll	0.12	0.12	50.0	50.0	50.0
Shoulder yaw	–	–	–	50.0	50.0
Elbow pitch	0.06	0.06	50.0	–	–
Elbow yaw	0.22	0.22	50.0	–	–
Elbow	–	–	–	50.0	50.0
Wrist pitch	–	50.0	–	–	50.0
Wrist yaw	–	50.0	–	–	50.0
Hand/wrist roll	–	50.0	–	50.0	50.0
Waist					
Waist yaw	5.0	5.0	–	50.0	50.0
Waist roll	–	–	–	–	50.0
Waist pitch	–	–	–	–	50.0
Left leg					
Hip pitch	0.01	0.01	0.01	0.01	0.01
Hip roll	1.0	1.0	1.0	1.0	1.0
Hip yaw	5.0	5.0	5.0	5.0	5.0
Knee	0.01	0.01	0.01	0.01	0.01
Ankle pitch	5.0	5.0	5.0	5.0	5.0
Ankle roll	5.0	5.0	5.0	5.0	5.0
Right leg					
Hip pitch	0.01	0.01	0.01	0.01	0.01
Hip roll	1.0	1.0	1.0	1.0	1.0
Hip yaw	5.0	5.0	5.0	5.0	5.0
Knee	0.01	0.01	0.01	0.01	0.01
Ankle pitch	5.0	5.0	5.0	5.0	5.0
Ankle roll	5.0	5.0	5.0	5.0	5.0