# Iterative Closed-Loop Motion Synthesis for Scaling the Capabilities of Humanoid Control

Supplementary Material

## Supplementary Material

This supplementary document provides additional details and results supporting the main paper *Iterative Closed-Loop Motion Synthesis for Scaling the Capabilities of Humanoid Control*. It is organized as follows:

- Sec. A: Notation and metric definitions used in Sec. 4 of the main paper.
- Sec. B: Complete description of the professional domain taxonomy and difficulty space (referencing Fig. 2 in the main paper).
- Sec. C: Prompt templates, variable libraries, and LLM prompting protocol used during iterative motion synthesis.
- Sec. D: Full motion generation and filtering pipeline, including MDM sampling settings and VLM-based semantic alignment.
- Sec. E: Complete training details for PHC and MaskedMimic controllers.
- Sec. F: Qualitative visualizations and failure case analyses.

## A. Notation and Metrics Definition

In this section we summarize the notations used throughout the paper and provide precise definitions of the tracking metrics reported in Sec. **??**. All metrics are computed on 3D human skeletons reconstructed in the global coordinate frame.

### A.1. Notation

A motion clip is represented as a sequence of $T$ frames with $J$ joints per frame. We use the following symbols:

- $t \in \{1, \dots, T\}$: time index;
- $j \in \{1, \dots, J\}$: joint index;
- $r$: index of the root joint (pelvis);
- $\mathbf{p}_t^j \in \mathbb{R}^3$: ground-truth global 3D position of joint $j$ at frame $t$;
- $\hat{\mathbf{p}}_t^j \in \mathbb{R}^3$: predicted global 3D position produced by the controller;
- $\mathbf{v}_t^j$, $\hat{\mathbf{v}}_t^j$: ground-truth / predicted joint velocities;
- $\mathbf{a}_t^j$, $\hat{\mathbf{a}}_t^j$: ground-truth / predicted joint accelerations.

Unless otherwise stated, we estimate velocities and accelerations by finite differences on the global joint trajectories:

$$\mathbf{v}_t^j = \mathbf{p}_{t+1}^j - \mathbf{p}_t^j, \tag{1}$$

$$\mathbf{a}_t^j = \mathbf{v}_{t+1}^j - \mathbf{v}_t^j, \tag{2}$$

and analogously for $\hat{\mathbf{v}}_t^j$ and $\hat{\mathbf{a}}_t^j$. For clips with frame interval $\Delta t \neq 1$, all velocities and accelerations are implicitly normalized by $\Delta t$ and $\Delta t^2$, respectively.

### A.2. Pose Error Metrics

We adopt standard pose tracking metrics and use the same notation throughout all experiments.

**Global MPJPE (g-MPJPE).** Global mean per-joint position error measures the average Euclidean distance between the predicted and ground-truth joint positions in the world coordinate frame:

$$\text{g-MPJPE} = \frac{1}{TJ} \sum_{t=1}^{T} \sum_{j=1}^{J} \left\| \mathbf{p}_t^j - \hat{\mathbf{p}}_t^j \right\|_2. \tag{3}$$

Lower values indicate better pose tracking accuracy.

**Local MPJPE (l-MPJPE).** Local MPJPE removes the root translation before computing the error, and focuses on the relative pose:

$$\text{l-MPJPE} = \frac{1}{TJ} \sum_{t=1}^{T} \sum_{j=1}^{J} \left\| (\mathbf{p}_t^j - \mathbf{p}_t^r) - (\hat{\mathbf{p}}_t^j - \hat{\mathbf{p}}_t^r) \right\|_2. \tag{4}$$

This metric is less sensitive to global drift and mainly reflects the articulation quality.

### A.3. Kinematic Consistency Metrics

To evaluate how well the controller preserves the temporal consistency of the reference motion, we measure discrepancies in joint velocities and accelerations.

**Velocity distance (VelDist).** The velocity distance is defined as

$$\text{VelDist} = \frac{1}{(T-1)J} \sum_{t=1}^{T-1} \sum_{j=1}^{J} \left\| \mathbf{v}_t^j - \hat{\mathbf{v}}_t^j \right\|_2. \tag{5}$$

It captures how closely the controller matches the dynamics (speed and direction changes) of each joint.

**Acceleration distance (AccDist).** Similarly, acceleration distance measures discrepancies in joint accelerations:

$$\text{AccDist} = \frac{1}{(T-2)J} \sum_{t=1}^{T-2} \sum_{j=1}^{J} \left\| \mathbf{a}_t^j - \hat{\mathbf{a}}_t^j \right\|_2. \tag{6}$$

This metric is sensitive to abrupt changes, and strongly correlates with motion smoothness and physical plausibility.

### A.4. Success Rate

In addition to continuous errors, we report a success rate following prior work on motion tracking controllers.

For each frame $t$ in a clip, we compute the global pose error

$$e_t = \frac{1}{J} \sum_{j=1}^{J} \left\| \mathbf{p}_t^j - \hat{\mathbf{p}}_t^j \right\|_2 .$$

A frame is considered successful if $e_t < \tau$, where $\tau$ is a fixed distance threshold (we use $\tau = 0.5\,\mathrm{m}$ in all experiments). The clip-level success rate is then

$$s_{\text{clip}} = \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}[e_t < \tau], \tag{7}$$

and the dataset-level success rate is the average of $s_{\text{clip}}$ over all clips in the dataset. This definition matches the "per-frame normalized averages (Avg, calculated by clip)" reported in Tab. **??**.

## B. Domain Taxonomy and Difficulty Space

This section elaborates on the motion domain taxonomy and the difficulty space used in Sec. 3.1 of the main paper. The taxonomy determines *what type of skills* a prompt is allowed to describe, while the difficulty space specifies *how hard* the resulting motion should be along several orthogonal axes. Together they provide a structured semantic prior that guides both text-to-motion generation and the iterative prompt optimization in Sec. C.

### B.1. Five Professional Motion Domains

We partition our prompt and motion space into five professional domains: martial arts, dance, combat, sport, and gymnastics. For each domain $d$, we identify a small set of representative sub-categories capturing its core technical patterns, as summarized in Tab. 1. These sub-categories were manually defined using domain knowledge.

Table 1. Professional domains and typical skill categories.

| Domain | Examples of sub-categories |
|---|---|
| Martial arts | strikes, high kicks, evasions, spinning attacks |
| Dance | leaps, turns, floorwork, partnering interactions |
| Combat | grappling, footwork, takedowns, clinch entries |
| Sport | sprinting, agility steps, rapid direction changes |
| Gymnastics | swings, rolls, tumbling, vault-related movements |

During prompt construction (Sec. C), each prompt is first assigned to one domain $d$, which restricts the variable and template choices to the corresponding domain-specific lists. This prevents unrealistic combinations and helps the motion diffusion model stay on a coherent professional manifold.

### B.2. Difficulty Components

Rather than using a single scalar difficulty label, we factor difficulty into four components that describe *what makes a motion hard*:

- **Base action**: underlying primitive skills.
- **Combo action**: how primitives are chained into longer sequences.
- **Detail**: technical nuance such as precise limb placement, posture control, and impact mechanics.
- **Speed rhythm**: overall tempo, burstiness, and rhythmic patterns.

Table 2 summarizes representative examples.

Table 2. Difficulty components and representative examples.

| Component | Example variables |
|---|---|
| Base action | kick, leap, roll, punch, turn |
| Combo action | "kick → spin → land", "roll → rise → leap" |
| Detail | "precise foot placement", "controlled arm extension" |
| Speed & rhythm | fast-twitch combos, syncopated rhythm, explosive bursts |

In our implementation, each component is instantiated by a curated list of phrases in the variable library. During prompt generation, we always select at least one element from each component, ensuring that every prompt makes an explicit commitment about the core skill, the combo structure, the execution details, and the temporal profile of the movement. In practice, these four components are implemented as slots in our domain-specific variable library used for prompt construction (Sec. C.1), which enables controllable sampling in the difficulty space.

### B.3. Difficulty Levels and Curriculum Progression

The four components above span a conceptual *difficulty space*. Intuitively, a motion becomes harder when: (i) the base action itself is advanced (e.g., aerial twist vs. basic kick), (ii) the combo chain is longer or involves more challenging transitions, (iii) the technical details require finer control or tighter balance margins, or (iv) the speed and rhythm demand higher power or more abrupt tempo changes.

We do not assign a fixed global difficulty index to each clip. Instead, difficulty is controlled at two stages:

- **Prompt design.** At generation time, we randomly construct variable sets. This gives us coarse-grained control

over the initial difficulty of Loop 0 prompts in each domain.

- **LLM-based refinement.** In later loops, Gemini-CoT receives tracker metrics and dual-VLM difficulty analyses (Sec. C) and adjusts the prompts in the difficulty space: when both the controller and VLMs deem a motion too easy, the scheduler tends to upgrade the combo component and the required detail (e.g., adding extra turns, aerial segments, or tighter balance constraints); when a motion is already difficult but unstable, the scheduler applies smaller shifts to improve difficulty.

This design creates a curriculum-like progression over loops (Loop 0→Loop 6): easy motions start with simpler base actions and short combos and are gradually enriched with more demanding details and rhythms, while already challenging motions are refined more conservatively. The domain taxonomy and difficulty space thus provide the semantic coordinates along which the LLM can *navigate* and optimize prompts, instead of relying on unstructured text edits.

# C. Prompt Templates and LLM Iteration

This section details the prompt design and LLM-based scheduler used in our closed-loop data–controller co-evolution (Sec. 3.5, Alg. 1). We first describe the variable-based template library that grounds motion descriptions in domain expertise, then present the random-prompt baseline, and finally explain the Gemini-CoT iteration procedure that turns tracker feedback and VLM scores into difficulty-aware prompt updates.

## C.1. Domain-Specific Variable Library

As discussed in Sec. B, we factor motion difficulty into four components: base action, combo action, detail, and speed & rhythm. In our implementation, these conceptual components are realized as four slots in a domain-specific variable library that is used for prompt construction.

For each domain $d$ (martial arts, dance, combat, sport, gymnastics), we maintain four lists `base_action`, `combo_action`, `detail`, and `speed_rhythm` . Each element in a list is a short, human-written phrase that instantiates the corresponding component for that domain. Typical entries include:

- domain-specific primitive skills in `base_action` (e.g., kicks, leaps, rolls, turns);
- multi-step chains in `combo_action` that specify a temporal order of primitives (e.g., "kick → spin → land", "roll → rise → leap");
- execution constraints in `detail` that highlight balance, posture, or impact mechanics;
- temporal patterns in `speed_rhythm` describing the overall tempo and intensity profile.

Across domains, each slot contains on the order of tens of phrases, providing a rich but still well-controlled combinatorial space. The lists are manually curated using domain knowledge so that every phrase is biomechanically meaningful and can be reliably parsed by the motion diffusion model.

During prompt generation (Sec. C.2), we first choose a domain $d$ and then sample one or more phrases from each of the four lists associated with $d$. These phrases are plugged into the sentence templates described next, yielding professional-style descriptions that make an explicit commitment about the core skill, the combo structure, the execution details, and the temporal profile of the motion, without requiring any additional free-form editing.

## C.2. Template Library for Professional Motion Descriptions

Given the variable library, we define a small but expressive set of domain-specific sentence templates that plug the four slots into grammatically coherent, professional-style descriptions. Each template implements the structure *who–does–what–how–with what timing*.

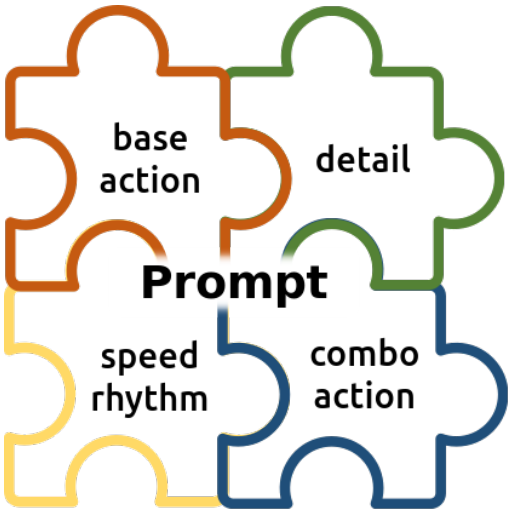



Figure 1. Structure of a prompt in our variable-based template design.

**Martial Arts**

```
``The martial artist executed
{combo_action} with {detail},
following {speed_rhythm}.''
``During training, the
practitioner performed
{base_action} focusing on {detail},
maintaining {speed_rhythm}.''
```

**Dance**

```
``The dancer performed
{combo_action} with {detail},
following {speed_rhythm}.''
``In the sequence climax, the
dancer executed {base_action}
followed by {combo_action} with
{detail}, using {speed_rhythm}.''
```

**Sports**

```
``The athlete performed
{combo_action} with {detail},
maintaining {speed_rhythm}.''
``During training, the
sportsperson executed {base_action}
focusing on {detail}, following
{speed_rhythm}.''
```

**Combat**

```
``The fighter launched
{combo_action} with {detail}, in
a {speed_rhythm} pattern.''
```

**Gymnastics**

```
``The gymnast completed
{combo_action} with {detail} under
{speed_rhythm}.''
```

These templates provide a consistent linguistic scaffold across domains, improving both MDM synthesis quality and VLM-based difficulty evaluation when filled with variables from Sec. C.1.

### C.3. Random Prompt Baseline

To disentangle the effects of professional templates from pure data scale, we also construct a purely random text baseline. This script calls a Gemini model with a simple instruction:

> *"Generate 200 random prompts for a Human Motion Diffusion Model. Each prompt must start with 'The person', be on a separate line, describe human movements, and have no duplicates."*

The resulting generic prompts (e.g., *"The person jumps and spins in the air before landing"*) do not use our variable library or templates. We synthesize motions from these random prompts and compare them, via t-SNE analysis and tracking performance, to motions generated from expert prompts and to professionally curated datasets. As discussed in the main paper, random-prompt motions tend to drift away from expert manifolds, highlighting the importance of structured, domain-aware prompt design.

### C.4. LLM-CoT Scheduler for Difficulty-Aware Prompt Updates

The core of our closed loop is an LLM-based scheduler. For each iteration $k$ and each domain, we group existing training clips into small batches of five "sets". For each set $i$, we collect:

- the original prompt $a_k^{(i)}$;
- the tracker metrics (success flag, global/local MPJPE, velocity and acceleration distances);
- dual VLM evaluations from GPT-4o and Qwen-VL-MAX: scalar difficulty scores in $[0, 10]$ and textual analyses of action sequence, technical complexity, movement intensity, and balance requirements.

We then construct a multi-turn Gemini-CoT conversation with four roles:

**(1) Core goal and optimization criteria.** A first message explains that the LLM should *"generate optimized prompts that push the motion tracker's performance limit"*, and must make every new prompt strictly harder than the original one. The message also specifies that the difficulty adjustment depends jointly on tracker metrics and VLM scores, e.g., actions with low errors and low VLM difficulty should receive a significant difficulty increase, while actions already hard for both tracker and VLM should only be slightly intensified.

**(2) Data summary for five sets.** A second message lists, for each set $i$, the original prompt, its tracking metrics, and the two VLM analyses with scores (0–10). This encodes the observation $o_k = [m_k, v_k, e_k]$ from Alg. 1 into a textual format that Gemini can parse.

**(3) Variable and template resources.** A third message exposes the domain-specific variable library and templates, instructing the LLM to use *only* these resources when generating new prompts. For example, for dance we provide the full lists of `base_action`, `combo_action`, `detail`, and `speed_rhythm`, together with the dance templates described above. This enforces that all optimized prompts remain on the professional motion manifold.

**(4) Chain-of-thought tasks and output format.** Finally, we specify explicit CoT tasks and an output schema:

- **Group analysis:** compare the five sets in terms of tracker metrics and VLM scores, and summarize agreement and disagreement between GPT-4o and Qwen-VL-MAX.
- **Per-set optimization:** for each set, explain (A) how the new prompt increases difficulty relative to the original,

(B) which variables are selected and why, (C) which template is used and why, and (D) output the final optimized prompt sentence.

This structured reasoning encourages the LLM to explicitly reason about where the tracker is weak or strong and how VLMs perceive difficulty before choosing variables and templates.

### C.5. Example: Dance Domain Optimization

To illustrate the behavior of the scheduler, we show a typical dance-domain batch (see the released logs for full text). For a set whose original prompt yielded low tracking error but mismatched VLM scores, the LLM selects a high-difficulty combo *"triple pirouette → aerial cartwheel → grand allegro"*, a demanding detail *"triple pirouette with back foot push, maintained turnout and steady rotation"*, and an intense rhythm *"leap sequence with slow preparation → explosive takeoff → suspended mid-air → quick landing"*. It then plugs these into a template of the form *"The sequence began with {combo_action}, then shifted into {base_action} with {detail}, regulated by {speed_rhythm}."*

For sets with already high VLM difficulty but moderate metrics, the LLM instead applies mild difficulty increases, e.g., by inserting a single extra high-leap base action while keeping the original combo and rhythm. Across domains, this policy realizes the qualitative logic of Alg. 1: prompts for easy motions are aggressively hardened, while already difficult motions are refined more conservatively, yielding a stable, curriculum-like progression from Loop 0 to Loop 6.

## D. Motion Generation and Filtering

### D.1. MDM Sampling Settings

Our setup is largely consistent with MDM-steps50-BERT, and we use the MDM-steps50-DistilBERT checkpoint. Additional details of our configuration are provided in Table 3.

Table 3. MDM Settings

| Parameters | Value |
|---|---|
| Data Source | HumanML3D |
| Diffusion Steps | 50 |
| Human Model | SMPL |
| Post-generation Joint Count | 22 |
| Post-modification Joint Count | 24 |
| Post-generation frame | 20 |
| Post-modification frame | 30 |
| Frames per Clip | 180 |
| Offset Height | 0.92 |

### D.2. Physical Validity Filtering

A generated clip is rejected if any of the following occur:

- root height outside a stable range,
- foot penetration below terrain plane,

Additionally, we perform temporal smoothing of the Root node's displacement offsets using a Gaussian filter to reduce motion noise.

### D.3. VLM Semantic Alignment

We encode each video as a uniformly subsampled sequence of keyframes stitched horizontally into a single Base64 image for the VLM, turning dynamic motion into a static yet temporally ordered trajectory that captures the full action while reducing compute versus raw video. For evaluation, we judge action semantics alignment by focusing on human motion cues in the stitched image and the prompt's action keywords and temporal logic while ignoring scene and prop differences, and we categorize results as aligned, partial, or mismatch using a permissive threshold. We render 60 frames per sequence and feed them to Qwen-VL-MAX. We obtain:

- action-matching score,
- difficulty score (1–10),
- domain classification.

A clip is accepted if:

$$\text{semantic\_score} \geq \tau_{\text{sem}}.$$

## E. Controller Training Details

### E.1. PHC Single-Primitive

Training parameters are as follows in Table 4:

Table 4. PHC Training Parameters

| Parameters | Value |
|---|---|
| Total Steps | $1.5 \times 10^6$ |
| Optimizer | Adam |
| Num Envs | 1024 |
| Learning Rate | $1 \times 10^{-4}$ |
| $\gamma$ | 0.98 |
| $\lambda$ | 0.95 |
| $\epsilon$ | 0.2 |
| Replay Buffer Size | $2 \times 10^5$ |

### E.2. MaskedMimic

Training parameters are as follows in Table 5:

## F. Qualitative Visualizations

We have supplemented the following visualization results(Fig. 2, Fig. 3a, Fig. 3b, Fig. 3c, Fig. 3d, Fig. 4, Fig. 5, Fig. 6).

Table 5. MaskedMimic Training Parameters

| Parameters | Value |
|---|---|
| Total Steps | $1.5 \times 10^6$ |
| Optimizer | Adam |
| Num Envs | 2048 |
| Learning Rate | $2 \times 10^{-5}$ |
| $\gamma$ | 0.99 |
| $\lambda$ | 0.95 |

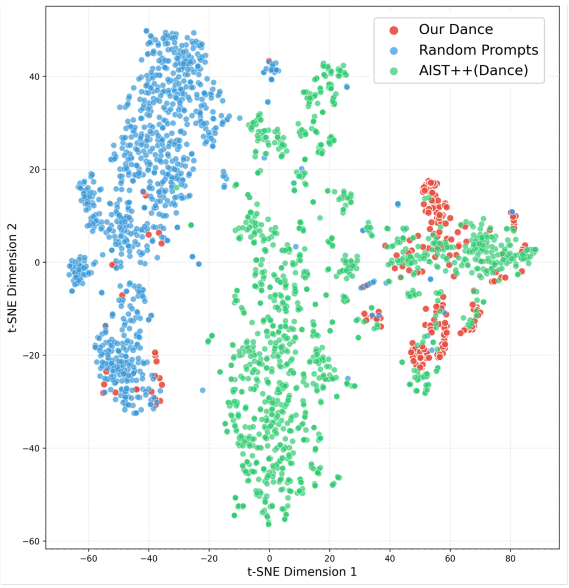
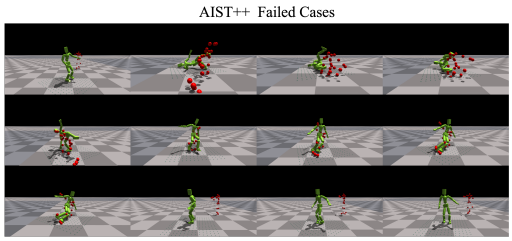
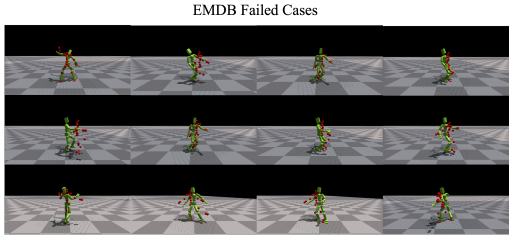


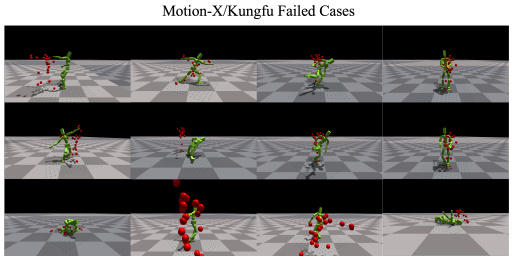

Figure 2. TSNE of Our Dance dataset and AIST++(Dance)



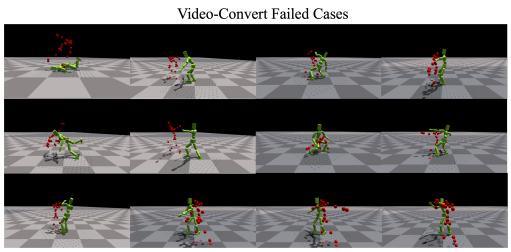

(a) Failure cases on AIST++




(b) Failure cases on EMDB




(c) Failure cases on Kungfu




(d) Failure cases on Video-Convert

Figure 3. Representative MaskedMimic failure cases on different datasets

Table 6. Performance Comparison on Different Datasets

| Metrics | Method \ Dataset | Loop0 | Loop1 | Loop2 | Loop3 | Loop4 | Loop5 | Loop6 | AMASS |
|---|---|---|---|---|---|---|---|---|---|
| **Success Rate ↑** | Kungfu | 37.9 | 47.7 | 51.7 | 59.1 | 55.8 | **60.6** | 60.3 | 47.1 |
| | EMDB | 31.1 | 33.3 | 51.1 | 64.4 | 55.6 | 60.0 | **64.4** | 53.3 |
| | AIST++ | 68.9 | 75.3 | 73.3 | 82.1 | 84.0 | 85.2 | **88.1** | 67.6 |
| | Video-Convert | 33.5 | 38.7 | 41.6 | 50.9 | 45.1 | 54.3 | **59.0** | 31.2 |
| **g-MPJPE(All) ↓** | Kungfu | 117.1 | 112.3 | 100.7 | 100.8 | **96.7** | 98.3 | 103.5 | 100.03 |
| | EMDB | 88.2 | 79.2 | 68.1 | 69.1 | 68.4 | **63.9** | 65.0 | 75.0 |
| | AIST++ | 105.5 | 98.8 | 97.7 | 94.2 | 89.2 | 87.8 | **86.1** | 101.5 |
| | Video-Convert | 136.3 | 134.9 | 126.2 | 122.2 | **118.8** | 121.3 | 121.3 | 132.6 |
| **Accel-dist(All) ↓** | Kungfu | 13.0 | 11.9 | 10.7 | **9.7** | 10.5 | **9.7** | 9.8 | 12.6 |
| | EMDB | 7.2 | 6.6 | 6.3 | 6.0 | **5.8** | **5.8** | **5.8** | 6.5 |
| | AIST++ | 12.3 | 11.8 | 11.6 | 10.8 | 11.1 | 10.3 | **10.2** | 14.5 |
| | Video-Convert | 17.2 | 16.1 | 15.6 | 14.9 | 15.3 | **14.1** | **14.1** | 21.0 |
| **Vel-dist(All) ↓** | Kungfu | 14.7 | 13.4 | 12.2 | **11.4** | 12.0 | **11.4** | **11.4** | 13.8 |
| | EMDB | 10.4 | 9.5 | 9.1 | 8.8 | 8.6 | **8.3** | **8.3** | 9.4 |
| | AIST++ | 16.4 | 15.4 | 15.2 | 14.5 | 14.7 | 14.0 | **13.8** | 17.5 |
| | Video-Convert | 21.0 | 20.0 | 19.2 | 18.3 | 18.6 | **17.6** | **17.6** | 23.6 |
| **g-MPJPE(Succ) ↓** | Kungfu | 74.9 | 72.4 | 67.3 | 68.9 | 66.2 | 71.4 | 73.2 | **63.4** |
| | EMDB | 69.7 | 60.0 | 57.6 | 63.2 | 62.2 | 57.7 | **54.8** | 63.0 |
| | AIST++ | 84.6 | 79.8 | 75.6 | 79.4 | **74.5** | 76.1 | 75.6 | 77.5 |
| | Video-Convert | 115.8 | 111.9 | 101.9 | 100.0 | **88.0** | 100.0 | 100.0 | 96.2 |
| **Accel-dist(Succ) ↓** | Kungfu | 3.2 | 3.3 | 3.5 | 3.6 | 3.7 | 3.8 | 3.8 | **3.1** |
| | EMDB | 4.8 | **4.2** | 4.5 | 5.0 | 4.8 | 4.6 | 4.4 | 4.6 |
| | AIST++ | 7.3 | 7.2 | **7.1** | 7.4 | 7.3 | 7.5 | 7.6 | 7.4 |
| | Video-Convert | 7.8 | 7.7 | 8.0 | 8.5 | **7.4** | 8.1 | 8.1 | 8.4 |
| **Vel-dist(Succ) ↓** | Kungfu | 6.2 | 6.1 | 6.0 | 6.1 | 6.3 | 6.4 | 6.2 | **5.3** |
| | EMDB | 8.2 | 7.1 | 7.3 | 7.9 | 7.6 | 7.4 | **7.0** | 7.6 |
| | AIST++ | 11.9 | 11.3 | **11.0** | 11.3 | 11.2 | 11.3 | 11.4 | **11.0** |
| | Video-Convert | 13.6 | 12.9 | 12.8 | 13.1 | **11.9** | 12.2 | 12.2 | 12.5 |

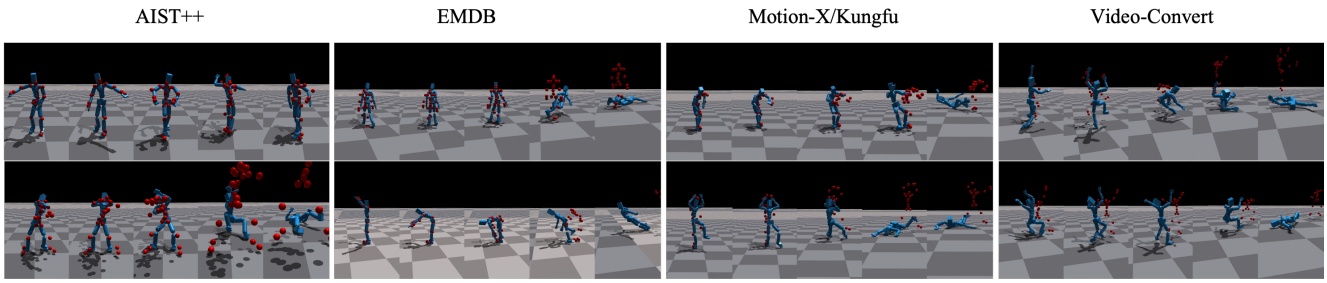



Figure 4. Representative Ours PHC failure cases.

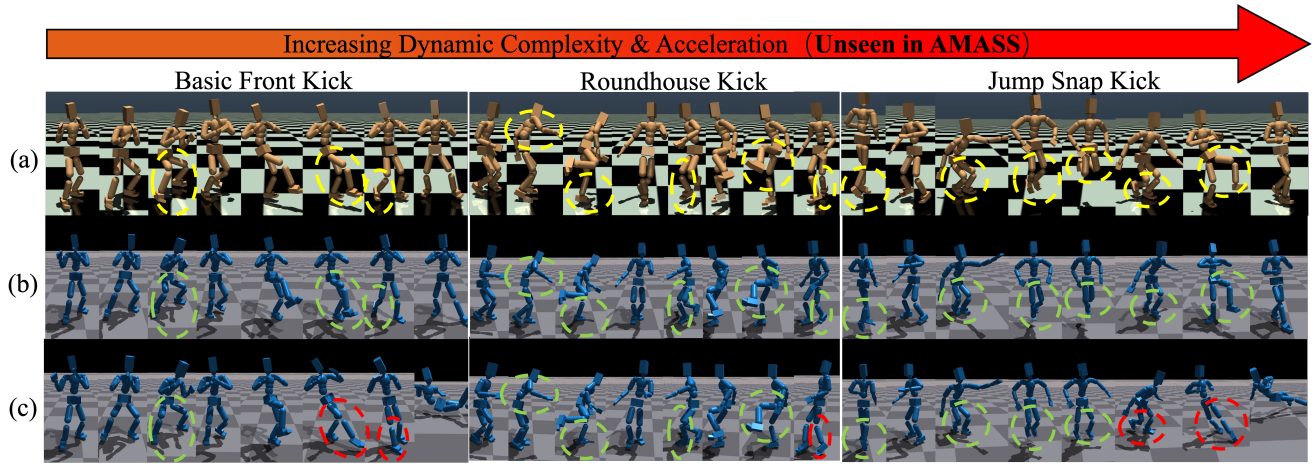



Figure 5. **Evolution of tracking capability** from initial stage ($L0$) to mastery ($L6$) across increasing dynamic complexity. (a) GT with yellow markers highlighting physically plausible anticipation (e.g., crouching before jumps). (b) Our final evolved $L6$ tracker achieves seamless mastery (green circles) as intensity scales. (c) The initial $L0$ tracker collapses (red circles) on high-acceleration maneuvers (e.g., Jump Snap Kick), highlighting the necessity of iterative co-evolution for capturing edge-case dynamics.

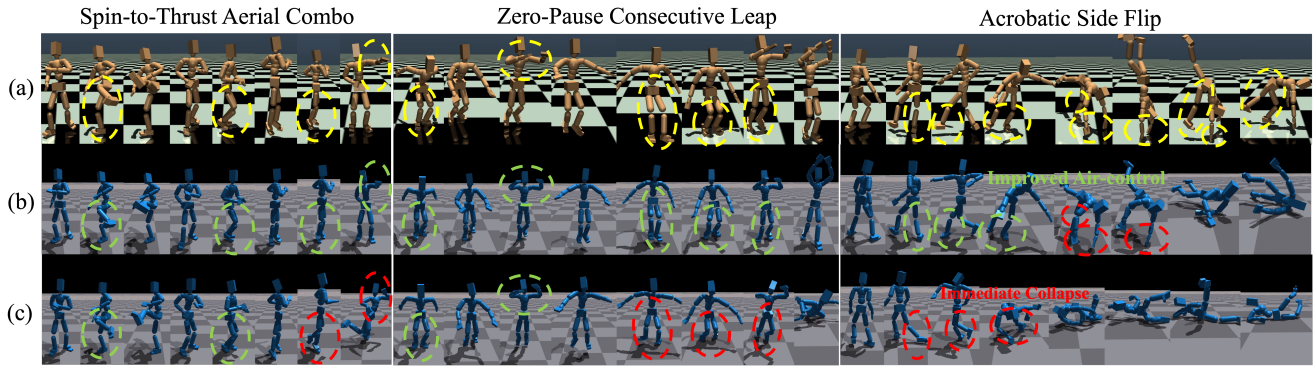



Figure 6. **Qualitative comparison on extreme OOD maneuvers** against the PHC baseline. (a) Reference GT motions exhibiting professional-grade agility. (b) Our $L6$ tracker maintains stable air-control and successful landing (green circles), even on the frontier "Acrobatic Side Flip". (c) The PHC baseline (trained on standard MoCap) suffers immediate collapse (red circles) when encountering the rapid momentum shifts and "Zero-Pause" transitions that define the dynamics gap.