# TEXEDO: Test Time Scaling for Controller-aware Language-conditioned Humanoid Motion Generation

Jianuo Cao*[1,2], Yuxin Chen*[2], Yuzhen Song[2,3], Masayoshi Tomizuka[2], Chenran Li[2], Thomas Tian[2]
[1]*Nanjing University* [2]*University of California, Berkeley* [3]*Southern University of Science and Technology*

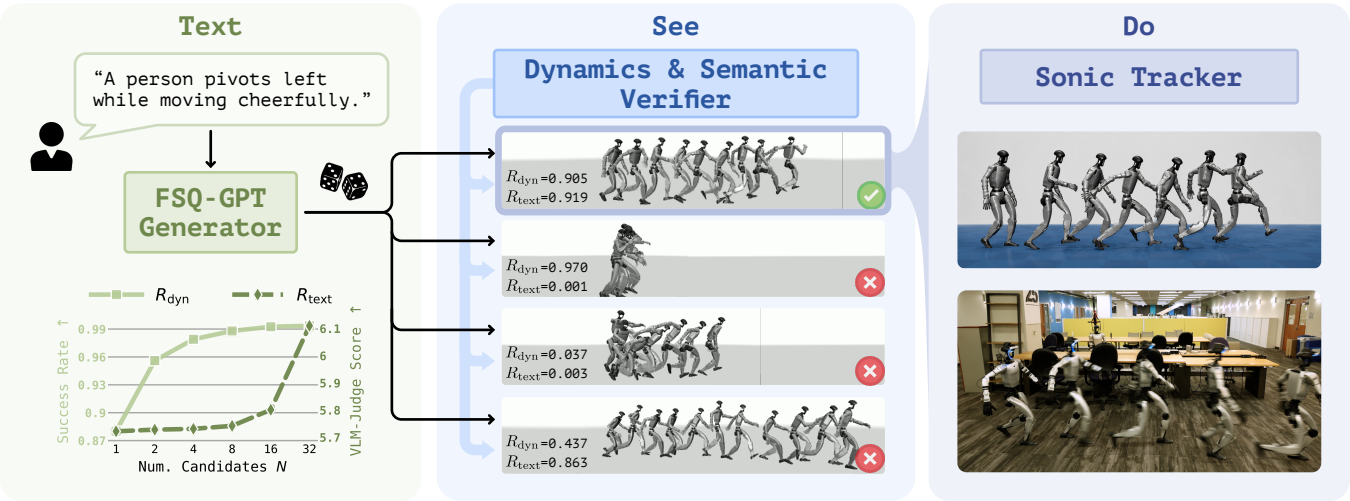


Fig. 1: **TEXEDO overview.** Given a language prompt, the TEXT stage samples multiple whole-body motions from a text-conditioned motion generator. The SEE stage evaluates each candidate with two grounded verifiers: a dynamic score that predicts controller executability and a semantic score that measures text-motion alignment. The DO stage selects a deployable motion by first filtering out dynamically infeasible candidates and then choosing the most semantically aligned one.

***Abstract*—Text-conditioned motion generation has become a promising interface for programming humanoid robots, but current generators are often trained on human motion datasets retargeted to robot morphology. While such data provides rich semantic kinematic priors, it does not capture the nuances of the whole-body tracking controller, including balance, contact, actuation limits, and controller-specific failure modes. Therefore, generated motions can be semantically plausible yet difficult or impossible for the robot to execute. We propose TEXEDO, a test-time scaling framework for humanoid motion generation that improves motion quality without requiring a stronger underlying generator. Given a text prompt, our method samples multiple motions from the pre-trained text-conditioned generator and selects the best executable and task-aligned motion. The reward model combines a dynamic feasibility verifier, distilled from whole-body tracking rollouts to predict physical executability, with a semantic alignment verifier that measures text-motion alignment in a learned co-embedding space. Our pipeline treats dynamic feasibility as a hard constraint and semantic alignment as the selection objective within the feasible set. Across large-scale simulation studies and real-world deployment on a Unitree G1, we show that our test-time scaling strategy consistently improves both tracking fidelity and text alignment, demonstrating that grounded verification is an effective path toward deployable language-guided humanoid motion generation. Please checkout our website for robot videos, code and data: Project Website.**

## I. INTRODUCTION

Recent progress on language-controlled humanoids has been a story of scaling at the two ends of a single deployment pipeline. On the *motion generation* side, text-conditioned motion generation models have grown from small task-specific networks into large models that produce whole-body motions directly from a language prompt [8, 24, 21, 6, 25, 18]. On the *whole-body control* side, learning-based whole-body tracking controllers have evolved from task-specific controllers into multi-task controllers that can, in principle, execute essentially any kinematic reference on real hardware [17, 12, 10, 13].

Despite this progress, these two scaling trends remain only weakly coupled. Most motion-generation models learn from human motion corpora, often after re-targeting the kinematics to a robot skeleton [14, 5], but their training objective does not expose them to the downstream controller that must realize the motion. As a result, a generated motion may be semantically faithful yet still fall outside the feasible set of the humanoid tracker due to balance constraints, contact timing, actuation limits, or controller-specific failure modes. In such cases, the burden is shifted to the controller: it must attempt to realize a reference motion that was never generated with its capabilities

in mind. This leads to brittle and poorly specified behavior at deployment, where the executed motion may deviate from the intended semantics, lose balance, produce improper contacts, or saturate actuators.

In language modeling, an increasingly successful way to improve generation at test time is to sample multiple candidates and use a verifier to select the best one [11, 19, 16, 7]. Recent work on vision-language-action policies suggests that a similar principle can improve robotic decision-making during deployment [9, 23]. However, importing this recipe to humanoid motion generation requires a more grounded notion of what "best" means. A good candidate motion must not only match the language prompt; it must also be executable by the particular whole-body controller on the particular humanoid platform. Thus, the verifier cannot merely judge abstract motion quality or text-motion similarity. It must be *controller-aware*: evaluate whether a candidate trajectory lies within the balance, contact, and actuation budgets of the tracker that will actually execute it.

We propose **TEXEDO** (**Text-See-Do**), a test-time scaling pipeline that turns a frozen text-to-motion generator into a controller-aware motion generator (Figure 1). Given a language prompt, TEXEDO first samples multiple candidate whole-body trajectories (TEXT). It then evaluates each candidate along two complementary axes (SEE): whether the motion is likely to be executable by the deployment-time tracker, and whether it remains semantically aligned with the language prompt. The first score is distilled from rollouts of the tracker itself, making it grounded in the balance, contact, and actuation limits of the target humanoid. The second score is learned from a contrastive text–motion embedding, allowing the selector to preserve the intended semantics of the prompt. Because dynamic feasibility and semantic alignment are not interchangeable, TEXEDO composes them asymmetrically (DO): it first removes motions that are unlikely to be trackable, and then selects the most semantically aligned candidate from the feasible set. In this way, TEXEDO converts additional inference-time samples into a single reference trajectory that is both language-aligned and grounded in the capabilities of the downstream controller, without modifying either the generator or the tracker.

The contributions of this work are threefold. First, we formulate controller-aware, language-conditioned whole-body humanoid motion generation as a test-time scaling problem, where additional inference-time samples are used to improve the single reference motion ultimately commanded to the robot. Second, we introduce two complementary grounded verifiers for controller-aware selection: a Dynamic Feasibility Verifier, distilled from rollouts of the deployment-time tracker to predict whether a reference motion can be realized on the target humanoid, and a Semantic Alignment Verifier, trained directly on robot-skeleton motions to measure text–motion alignment. Third, we show that semantic alignment and dynamic feasibility define competing selection criteria, and propose an asymmetric filter-then-rerank strategy that treats feasibility as a constraint before selecting the most semantically aligned motion. Empirically, TEXEDO improves both motion execution quality and task alignment, transfers zero-shot to unseen motion generators, generalizes to out-of-distribution prompts, and yields consistent gains over vanilla sampling without selection in real-world experiments on a physical Unitree G1 humanoid.

## II. PROBLEM FORMULATION

We formulate controller-aware whole-body motion generation as a *test-time scaling* problem: instead of committing to the first motion sampled from a generator, we spend inference-time compute to sample multiple candidate motions and select the one most suitable for execution. Formally, let $\ell$ denote a natural-language instruction and let $\mathbf{m} \in \mathbb{R}^{T \times D}$ denote a whole-body reference motion, where $T$ is the sequence length and $D$ is the dimension of the target humanoid's root pose and joint configuration. A text-to-motion generator $\mathcal{G}$ induces a distribution $\mathcal{G}(\cdot \mid \ell)$ over such motions. At deployment, rather than executing a single sample from this distribution, we draw a candidate set and use a selector to choose a motion that is both dynamically feasible for the downstream tracker and semantically aligned with the instruction.

## III. METHOD

Given a language instruction $\ell$, TEXEDO follows a TEXT–SEE–DO runtime selection pipeline. Section III-A (TEXT): A language-conditioned generator samples $N$ candidate motions $\{\mathbf{m}_i\}_{i=1}^{N}$ from $\ell$; Section III-B (SEE): grounded verifiers score their dynamic feasibility and semantic alignment; and Section III-C (DO): a filter-then-rerank rule selects one deployable motion. Both the generator and deployment-time tracker remain frozen.

### A. Language-Conditioned Motion Generator (TEXT)

The TEXT stage provides the candidate motions that the subsequent SEE and DO stages operate on. Given a language instruction $\ell$, we assume access to a language-conditioned motion generator $\mathcal{G}$ that can sample a set of candidate reference trajectories $\mathcal{M}_N(\ell) = \{\mathbf{m}_i\}_{i=1}^{N}, \mathbf{m}_i \sim \mathcal{G}(\cdot \mid \ell)$. In our framework, $\mathcal{G}$ is treated as a black box: the verifiers consume only the decoded reference trajectories $\{\mathbf{m}_i\}$ and do not rely on the generator architecture.

For the main experiments, we instantiate $\mathcal{G}$ with FSQ-GPT, a discrete autoregressive text-to-motion generator for whole-body humanoid trajectories. FSQ-GPT first compresses continuous robot motions into discrete motion tokens using an FSQ tokenizer [15], and then fine-tunes a Flan-T5-base model [4] to generate these tokens autoregressively from language instructions. At inference time, sampled token sequences are decoded back into continuous whole-body reference trajectories, forming the candidate set $\mathcal{M}_N(\ell)$. We sample with temperature and top-$k$ truncation to control candidate diversity. Full training and sample details for FSQ-GPT are provided in Appendix B.

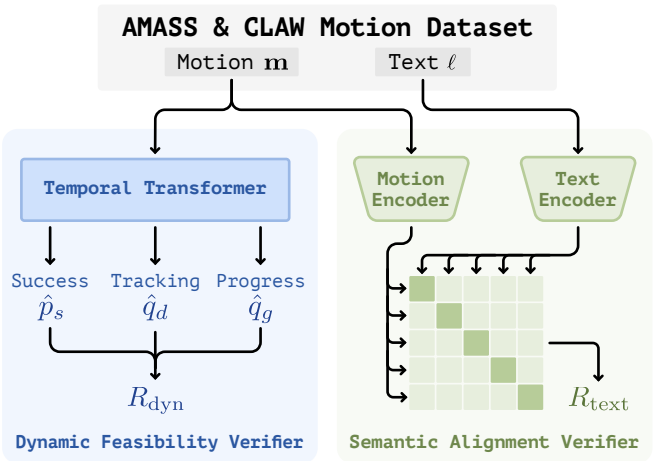


Fig. 2: **Dual verifier design.** $R_{\text{dyn}}$ estimates dynamic feasibility from the motion alone and $R_{\text{text}}$ measures text-motion alignment in a learned embedding space.

### *B. Dynamic Feasibility And Semantic Alignment Evaluation* (SEE)

What turns a pool of candidate motions into a single deployable trajectory is, ultimately, a definition of *motion quality*. The SEE stage answers this question with two scalars assigned to every candidate: a dynamic score $R_{\text{dyn}}(\mathbf{m}) \in [0,1]$ measuring whether the motion is dynamically feasible on the deployment-time tracker, and a semantic score $R_{\text{text}}(\ell, \mathbf{m}) \in [0,1]$ measuring whether it depicts what the prompt describes. The design of both verifiers is illustrated in Figure 2.

**Dynamic feasibility verifier.** A kinematically plausible motion may still fail under the deployment-time tracker because of balance, contact, or actuation constraints. However, evaluating each candidate by an actual roll-out at selection time is prohibitive. Thus, we proceed in two steps:

*Oracle quality of roll-outs.* For each candidate $\mathbf{m}$, an offline tracker roll-out produces three complementary signals: a binary success indicator $y_s \in \{0,1\}$, a normalized tracking quality score $q_d \in [0,1]$, and a progress ratio $q_g \in [0,1]$ measuring the percentage of reference completed before early termination. We adopt the early-termination criterion of BeyondMimic [10] as the common failure definition for both $y_s$ and $q_g$, and details are reported to Appendix C.

To jointly capture these signals, we collapse the three signals into a single oracle quality:

$$Q^*(y_s, q_d, q_g) \;\triangleq\; y_s \cdot \frac{1+\alpha\, q_d}{1+\alpha} \;+\; (1-y_s)\,\beta\, q_g\, q_d. \qquad (1)$$

Here, $\alpha$ controls how much motion-tracking quality modulates the reward for a *successful* roll-out, while $\beta$ sets the partial-credit weight assigned to *failed* roll-outs based on their progress and tracking quality. In particular, we set $\beta < 1/(1+\alpha)$, so that $Q^*$ assigns successful roll-outs a higher score than any failed roll-outs, while retaining the graded information within the all success or all failure group.

*Motion-only surrogate.* After obtaining the oracle quality labels, we train a lightweight temporal Transformer to predict $(\hat{p}_s, \hat{q}_d, \hat{q}_g)$ from $\mathbf{m}$, and feed the predictions back through Eq. (1):

$$R_{\text{dyn}}(\mathbf{m}) \;\triangleq\; Q^*\big(\hat{p}_s(\mathbf{m}),\, \hat{q}_d(\mathbf{m}),\, \hat{q}_g(\mathbf{m})\big), \qquad (2)$$

the surrogate and its supervision signal share exactly one algebraic form. The three heads are trained jointly under a weighted sum of a positive-balanced binary cross-entropy on $\hat{p}_s$, an MSE on $\hat{q}_d$, and an MSE on $\hat{q}_g$ masked to failed rollouts. Full architecture and training details are in Appendix C.

**Semantic alignment verifier.** While dynamic realizability is essential, a motion is only successful if it also realizes the semantic intent of the text instruction. Following the T2M [5], we train bidirectional GRU text and motion encoders, $\varphi_{\text{text}}$ and $\varphi_{\text{motion}}$, *directly* on the target robot skeleton, so that the embedding space is consistent with the candidates produced by the TEXT stage. With $d_{ij} = \|\varphi_{\text{text}}(\ell_i) - \varphi_{\text{motion}}(\mathbf{m}_j)\|_2$, we optimize an all-pairs margin contrastive objective:

$$\mathcal{L}_{\text{match}} \;=\; \frac{1}{B}\sum_i d_{ii}^2 + \frac{1}{B(B-1)}\sum_{i\neq j}\big[\delta - d_{ij}\big]_+^2, \qquad \delta = 2.0, \qquad (3)$$

where $[x]_+ = \max(0,x)$ and $B$ is batch size. The all-pairs term exposes the encoders to $\mathcal{O}(B^2)$ informative negatives per step rather than the single random negative used in the original T2M formulation. At test time, the semantic score is defined as the exponentiated negative embedding distance:

$$R_{\text{text}}(t,m) \triangleq \exp\big(-\|\phi_{\text{text}}(t) - \phi_{\text{motion}}(m)\|_2\big) \in (0,1], \qquad (4)$$

so a higher $R_{\text{text}}$ indicates better text–motion alignment.

### *C. Test-Time Selection* (DO)

This stage converts the verifier scores into a single deployable reference motion. Given a prompt $\ell$ and candidate motions $\{\mathbf{m}_i\}_{i=1}^N$, we first filter candidates predicted to be dynamically infeasible and then select the most semantically aligned survivor:

$$\hat{\mathcal{S}} = \big\{\mathbf{m}_i : R_{\text{dyn}}(\mathbf{m}_i) > \theta\big\},$$
$$\mathbf{m}^* = \begin{cases} \arg\max\limits_{\mathbf{m}_i \in \hat{\mathcal{S}}} R_{\text{text}}(\ell, \mathbf{m}_i), & \hat{\mathcal{S}} \neq \varnothing, \\ \arg\max\limits_{i} R_{\text{dyn}}(\mathbf{m}_i), & \text{otherwise.} \end{cases} \qquad (5)$$

The rule here is deliberate and reflects a deployment-safety priority: on physical hardware, a semantically perfect but dynamically infeasible motion can cause falls or actuator saturation, whereas a feasible but slightly less-aligned motion merely underperforms. We therefore treat dynamic feasibility as a hard constraint to be satisfied first, and semantic alignment as the objective to optimize only within the executable pool. We set $\theta = 0.8$ from validation roll-outs and the empty-set fallback $\hat{\mathcal{S}} = \varnothing$ is triggered on only $1.18\%$ of test prompts.

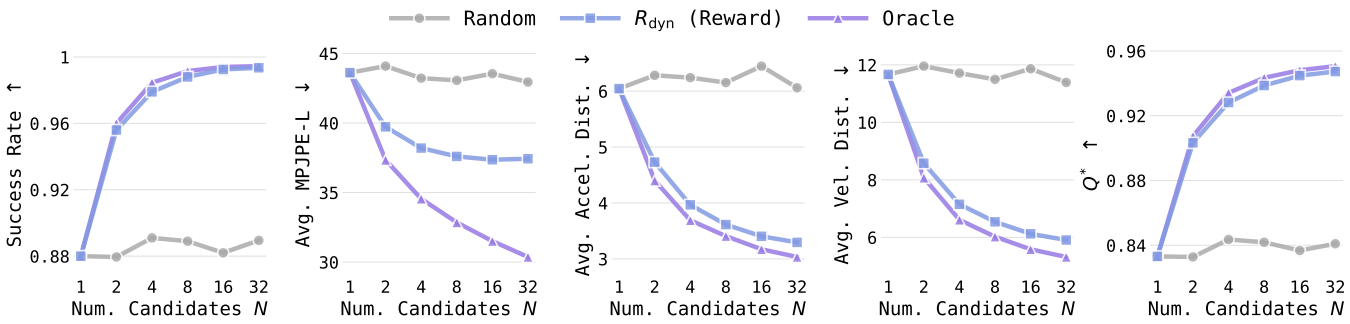


Fig. 3: **Best-of-$N$ curves for dynamic feasibility on FSQ-GPT.** Subplots report Succ, $E_{\text{mpjpe-l}}$, $E_{\text{acc}}$, $E_{\text{vel}}$, and $Q^*$ as $N$ increases from 1 to 32. Across metrics, $R_{\text{dyn}}$**-only** consistently outperforms **Random** and closely approaches **Oracle**.

## IV. Experiments

We evaluate TEXEDO as a deployable test-time scaling system for language-conditioned humanoid motion generation along three axes: test-time selection performance, zero-shot generalization, and real-world transfer. **Q1.** Do the grounded verifiers and their composition select motions that are both executable by the whole-body controller and aligned with the language instruction (Section IV-B)? **Q2.** Can the verifiers transfer plug-and-play to an unseen generator and out-of-distribution prompts without retraining? (Section IV-C)? **Q3.** Do the simulation gains carry over to real-world execution on a Unitree G1 humanoid (Section IV-D)?

### A. Experimental Setup

**Robot set up and model training.** We use a Unitree G1 humanoid as our robot platform with SONIC [13] as the frozen whole-body controller. We train our text-to-motion generator using AMASS [14] and CLAW [2]. More implementation details can be found in Appendix B.

**Baselines.** We compare **TEXEDO** against four selection strategies: (i) **Base** ($N$=1), a single sample from the generator (no-test-time-scaling); (ii) $R_{\text{dyn}}$**-only**, $\arg\max_i R_{\text{dyn}}(\mathbf{m}_i)$, a purely physical selector; (iii) $R_{\text{text}}$**-only**, $\arg\max_i R_{\text{text}}(\ell, \mathbf{m}_i)$, a purely semantic selector; and (iv) **Oracle**, a privileged selector that rolls out *every* candidate in $\mathcal{M}_N(\ell)$ through SONIC and picks the one that actually scores best on each metric. Note that **Oracle** requires $N$ SONIC rollouts per prompt (three orders of magnitude more expensive than scoring with the verifiers) and is therefore impractical at deployment time; we report it only as an upper bound on any selector over the same pool.

**Evaluation metrics.** We evaluate selected motions along two complementary dimensions. *Dynamic fidelity* measures whether a reference motion can be faithfully executed by the downstream tracker. We report tracking success rate (Succ), root-relative mean per-joint position error ($E_{\text{mpjpe}}$, mm), joint acceleration error ($E_{\text{acc}}$, mm/frame$^2$), joint velocity error ($E_{\text{vel}}$, mm/frame), and the composite oracle quality $Q^*$ defined in Equation (1). *Semantic alignment* measures whether the selected motion matches the language instruction, using VLM-as-Judge as our primary evaluation. Full details of metric definitions and the VLM-based evaluation method are provided in the Appendix E.

**Inference-time budget.** All timings are measured on a single NVIDIA A100-SXM4 40GB GPU. Because the $N$ candidates for a prompt are sampled and scored as a single batch, the cost of test-time scaling grows sublinearly in $N$. Sampling 32 candidates from FSQ-GPT takes 2.62 s on average, and scoring 32 candidates adds only 15.12 ms (Dynamic Feasibility Verifier) and 11.93 ms (Semantic Alignment Verifier) on average, so a full $N$=32 sample and selection completes in roughly 3.5 s end-to-end.

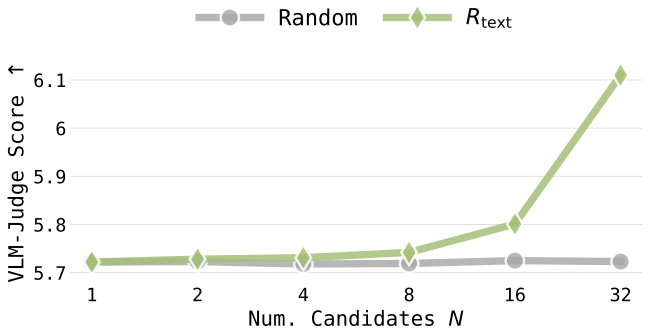


Fig. 4: **Semantic alignment verifier turns larger candidate pools into better prompt alignment.** For each prompt, $R_{\text{text}}$**-only** selects the candidate with the highest $R_{\text{text}}$ score; the selected motions improve consistently with increasing $N$ when evaluated by an independent VLM Judge.

### B. Effectiveness Of TEXEDO For Test-Time Scaling

**Effectiveness of individual verifiers.** We first isolate the effect of each verifier under the same best-of-$N$ setting. For each prompt, we sample a candidate pool of $N$ different samples and select one motion using either the Dynamic Feasibility Verifier or the Semantic Alignment Verifier alone.

Figure 3 reports the dynamic feasibility metrics of the selected motions as the candidate budget $N$ increases, comparing $R_{\text{dyn}}$**-only** against **Random** and **Oracle**. As $N$ increases,

TABLE 1: **TEXEDO balances the complementary strengths of $R_{\text{dyn}}$ and $R_{\text{text}}$.** Pushing either verifier to its extreme costs the other, and TEXEDO achieves strong performance on both axes simultaneously.

| Strategy | Semantic alignment | Motion execution quality | | | | |
|---|---|---|---|---|---|---|
| | VLM-Judge↑ | Succ↑ | $E_{\text{mpjpe}}$ ↓ | $E_{\text{acc}}$ ↓ | $E_{\text{vel}}$ ↓ | $Q^*$ ↑ |
| **Base** ($N$=1) | 5.722 | 0.873 | 44.34 | 6.09 | 11.82 | 0.829 |
| $R_{\text{dyn}}$**-only** ($N$=32) | 4.924 | **0.990** | **38.15** | **3.30** | **5.91** | **0.945** |
| $R_{\text{text}}$**-only** ($N$=32) | **6.110** | 0.885 | 42.97 | 4.90 | 10.03 | 0.847 |
| **TEXEDO** ($N$=32) | 6.054 | 0.984 | 39.09 | 4.26 | 7.78 | 0.926 |

TABLE 2: **Zero-shot transfer to Kimodo.** TEXEDO, trained only on FSQ-GPT rollouts, improves both VLM-Judge and physical execution metrics when applied zero-shot to the unseen Kimodo motion generator.

| Method | Semantic alignment | Motion execution quality | | | | |
|---|---|---|---|---|---|---|
| | VLM-Judge↑ | Succ↑ | $E_{\text{mpjpe}}$ ↓ | $E_{\text{acc}}$ ↓ | $E_{\text{vel}}$ ↓ | $Q^*$ ↑ |
| Kimodo (**Base**, $N$=1) | 4.823 | 0.937 | 38.62 | 1.84 | 5.33 | 0.918 |
| Kimodo+$R_{\text{dyn}}$**-only** ($N$=32) | 5.020 | **0.955** | **35.47** | **1.42** | **4.28** | **0.937** |
| Kimodo+$R_{\text{text}}$**-only**($N$=32) | **5.717** | 0.942 | 38.02 | 1.87 | 5.43 | 0.919 |
| Kimodo+**TEXEDO** ($N$=32) | 5.381 | 0.954 | 37.49 | 1.70 | 4.97 | 0.935 |

$R_{\text{dyn}}$**-only** consistently outperforms **Random** and approaches **Oracle** across physical metrics, showing that the learned verifier recovers much of the improvement available from oracle-based selection. This is important because **Oracle** requires rolling out every candidate with high-fidelity simulation, making it impractical as an online selection rule.

A similar trend holds for semantic alignment. When selecting candidates by $R_{\text{text}}$, the VLM-as-Judge score (independent third-party judge) increases monotonically with the candidate budget, from 5.722 at $N$=1 to 6.110 at $N$=32 (Figure 4).

Together, these results show that the two verifiers convert additional test-time compute budgets into improvements along their intended axes: $R_{\text{dyn}}$ improves executability, while $R_{\text{text}}$ improves semantic alignment.

**Effectiveness of TEXEDO.** We next evaluate the full selector, which composes the two verifiers through an asymmetric filter-then-rerank rule. Given a candidate pool, we first filter out motions predicted to be dynamically infeasible, and then selects the most semantically aligned motion among the remaining candidates.

Table 1 compares TEXEDO against **Base** and two single-verifier selectors ($R_{\text{dyn}}$**-only** and $R_{\text{text}}$**-only**) at $N$=32. Optimizing feasibility alone is not neutral to semantics but detrimental: $R_{\text{dyn}}$**-only** drives the VLM-Judge score down to 4.924, *below* the no-selection **Base** (5.722), as the most trackable candidates tend to be conservative motions that drift from the prompt's intent. Symmetrically, $R_{\text{text}}$**-only** reaches 6.110 in alignment but yields little improvement in execution (Succ 0.885 vs. **Base**'s 0.873, far below $R_{\text{dyn}}$**-only**'s 0.990). Since pushing either verifier to its extreme costs the other, TEXEDO resolves the tension by composition rather than compromise: it recovers execution close to $R_{\text{dyn}}$**-only** (Succ 0.984 vs. 0.990) while largely preserving the semantic alignment of $R_{\text{text}}$**-only** (6.054 vs. 6.110).

**Qualitative comparison.** Figure 5 shows four representative case studies, illustrating different failure modes of the baselines and a successful case of TEXEDO.

### C. Out-Of-Distribution Generalization

**Zero-shot transfer to an unseen motion generator.** A key advantage of TEXEDO is that its verifiers operate on decoded robot motions rather than generator-specific internals. The Dynamic Feasibility Verifier consumes only the candidate reference motion, and the Semantic Alignment Verifier consumes only the motion and the language instruction. As a result, the selection rule is not tied to the architecture, tokenization scheme, or latent space of the generator used to produce the candidates.

To test this generator-agnostic property, we apply the same verifiers trained on FSQ-GPT rollouts directly to Kimodo [18], an independently trained motion generator zero-shot. As shown in Table 2, TEXEDO improves both execution quality and semantic alignment on this unseen generator, demonstrating that the learned criteria transfer beyond the generator distribution used to train the verifiers.

**Out-of-distribution (OOD) generalization.** We further test whether TEXEDO remains effective on language instructions outside the training distribution. The base generator is trained on the combined AMASS [14] and CLAW [2] datasets, whereas the OOD evaluation uses 50 prompts sampled from BONES-SEED [1]. This setting tests whether the verifier-based selection rule can improve generation when the prompt composition differs from the data used to train the generator and verifiers. As shown in Table 3, TEXEDO improves over the base generator on both ID and OOD prompts along both axes. On the OOD split, it raises tracking success from 0.860

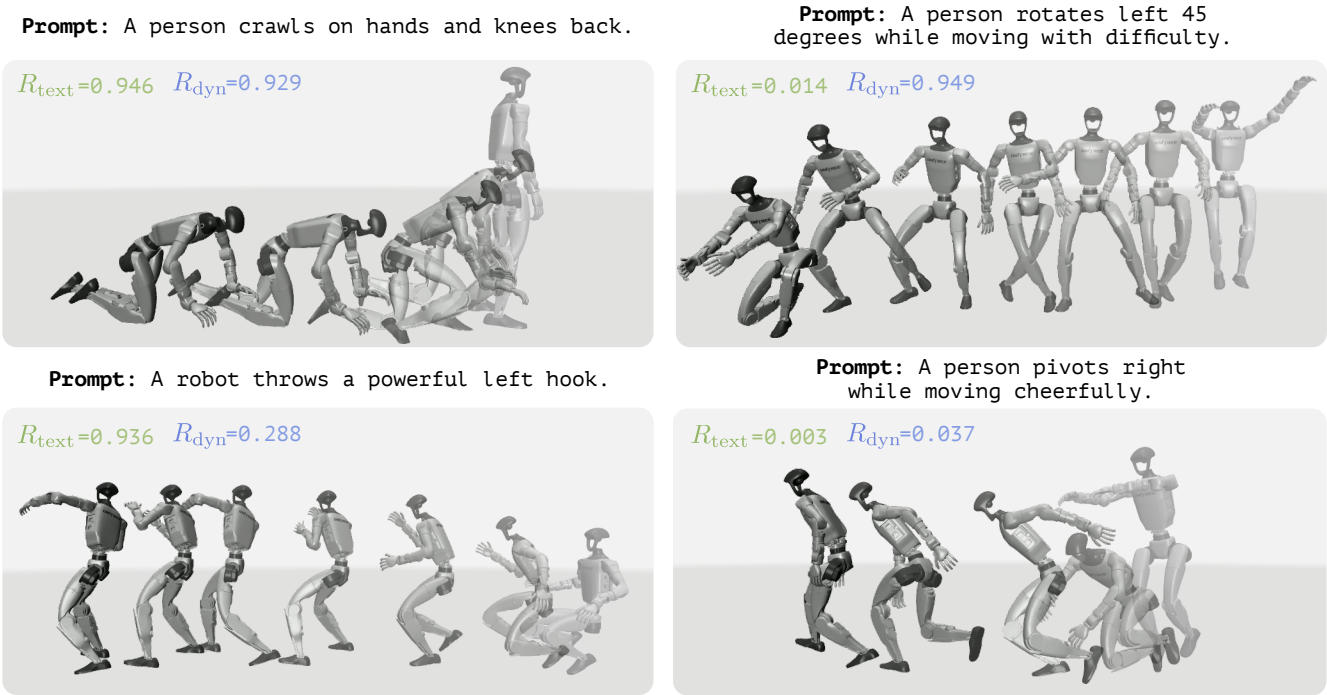


Fig. 5: **Qualitative comparison.** Representative case studies in MuJoCo simulation. **Top-left:** a success case of TEXEDO, where the generated motion is dynamically feasible and semantically aligned. **Top-right:** a failure case of $R_{\text{dyn}}$**-only** produces a stable but semantically generic motion. **Bottom-left:** a failure case of $R_{\text{text}}$**-only** selects a semantically aligned but untrackable motion. **Bottom-right:** a failure case of **Base** fails with a dynamically infeasible and semantically inconsistent motion.

TABLE 3: **Out-of-distribution generalization.** Evaluated on prompts from unseen dataset BONES-SEED [1], TEXEDO achieves a strong balance between dynamics fidelity and semantic alignment.

| Setting | Method | Semantic alignment | Motion execution quality | | | | |
|---|---|---|---|---|---|---|---|
| | | VLM-Judge↑ | Succ↑ | $E_{\text{mpjpe}}$ ↓ | $E_{\text{acc}}$ ↓ | $E_{\text{vel}}$ ↓ | $Q^*$ ↑ |
| ID | Base($N$=1) | 5.722 | 0.873 | 44.34 | 6.09 | 11.82 | 0.829 |
| | TEXEDO ($N$=32) | **6.054** | **0.984** | **39.09** | **4.26** | **7.78** | **0.926** |
| OOD | Base($N$=1) | 4.116 | 0.860 | 36.22 | 8.36 | 15.71 | 0.804 |
| | TEXEDO ($N$=32) | **4.401** | **0.980** | **26.64** | **3.65** | **3.88** | **0.942** |

to 0.980 and oracle quality $Q^*$ from 0.804 to 0.943 and substantially reducing tracking errors. These results indicate that the feasibility verifier generalizes well beyond the training distribution. In contrast, the semantic gain is smaller than on ID prompts, likely because the learned semantic embedding transfers less readily to unseen prompts.

### *D. Real-Robot Deployment*

We deploy TEXEDO on a physical Unitree G1 humanoid across 30 text prompts spanning locomotion, upper-body gestures, and their combinations. Using N=32 candidates per prompt, TEXEDO successfully executes all 30 prompts on hardware. Figure 6 shows representative qualitative results: TEXEDO consistently selects motions that execute stably and match the intended behavior. Full deployment details including hardware setup, prompt construction, and per-prompt tracking results are provided in Appendix D-0b.

## V. CONCLUSION

In this work, we presented TEXEDO, a test-time scaling framework that turns a frozen language-conditioned motion generator into a controller-aware humanoid motion system. By sampling multiple candidate motions and selecting among them with two complementary verifiers for dynamic feasibility and semantic alignment, TEXEDO improves the single reference ultimately commanded to the whole-body tracker without retraining either component.

Our evaluation shows that the verifiers are effective along their intended axes: the Dynamic Feasibility Verifier selects motions that are more executable by the downstream tracker, while the Semantic Alignment Verifier selects motions that

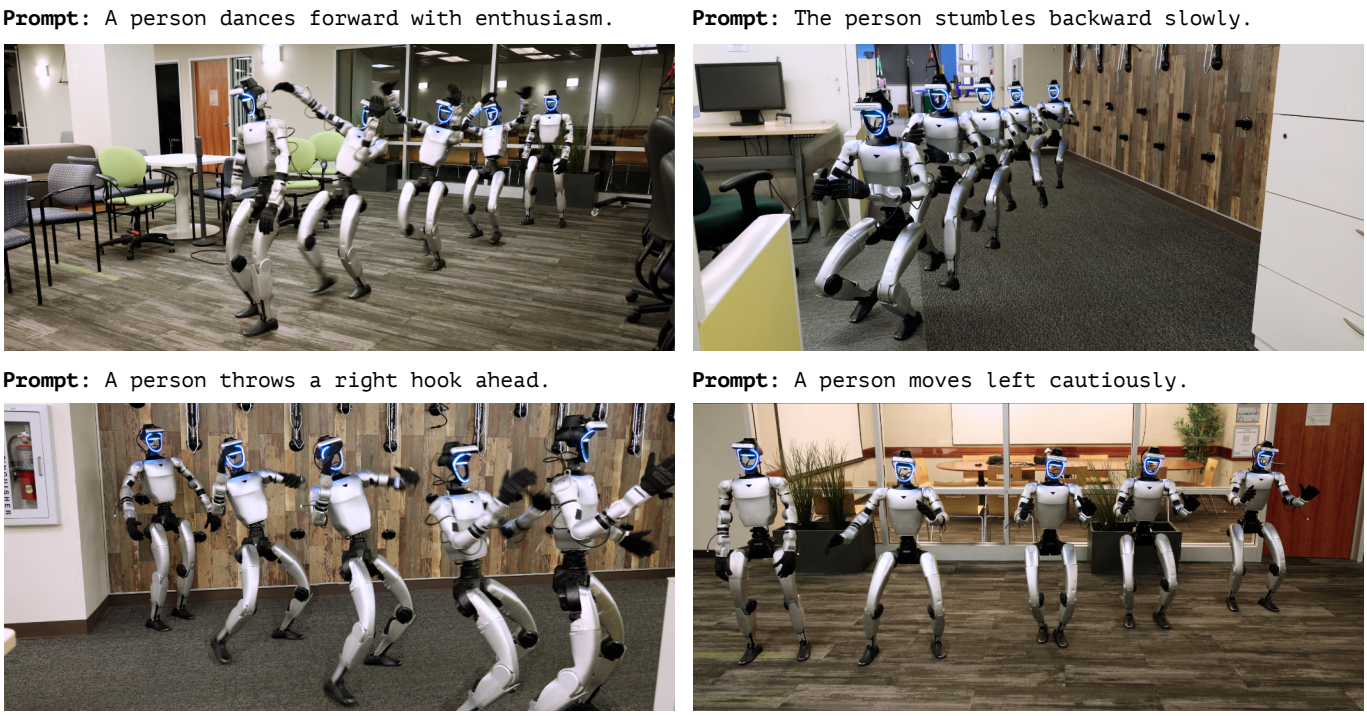


Fig. 6: **Real-robot execution results on the Unitree G1.** Across diverse prompt types spanning locomotion, upper-body gestures, and their combinations, TEXEDO consistently selects motions that execute stably and faithfully realize the intended semantics on physical hardware.

better preserve the language instruction. The proposed filter-then-rerank rule combines these complementary signals to select motions that improve both execution quality and semantic alignment. The same verifiers transfer plug-and-play to an unseen generator and out-of-distribution prompts without retraining, and the resulting gains carry over to real-world execution on a Unitree G1 humanoid. These findings demonstrate the effectiveness of the proposed method for improving deployable language-guided humanoid motion generation.

## VI. Limitations

**Candidate-set and latency trade-off.** As a test-time selection framework, TEXEDO is bounded by the quality and diversity of the sampled candidate motions. If the generator rarely produces motions that are both semantically aligned and dynamically feasible, or if the tracker is too weak to realize any motions, selection alone cannot recover a successful motion. Increasing the number of samples $N$ can improve the chance of finding a valid candidate, but this introduces an inference-time trade-off between selection quality and latency; future work could address this with adaptive sampling mechanisms that allocate additional samples only when needed.

**Verifier data and tracker adaptation.** The grounded verifiers need diverse reference motions and tracker roll-outs to learn reliable semantic and dynamic scores. Moreover, because dynamic feasibility is tracker-specific, changing the tracker requires rerunning roll-outs to update oracle feasibility labels and adapt the Dynamic Verifier. However, the cost of this adaptation process can be effectively mitigated through a data reuse mechanism: the reference-motion corpus is not tied to a particular generator, and even under tracker changes the same references can be reused for relabeling, which is substantially less costly than collecting a new motion corpus or retraining the motion generator. Future work could further reduce this cost by investigating more efficient fine-tuning procedures for verifier adaptation.

## References

[1] Bones Studio. BONES-SEED: Skeletal everyday embodiment dataset, 2026. URL https://huggingface.co/datasets/bones-studio/seed.

[2] Jianuo Cao, Yuxin Chen, and Masayoshi Tomizuka. Claw: Composable language-annotated whole-body motion generation. *arXiv preprint arXiv:2604.11251*, 2026.

[3] Yuxin Chen, Jianglan Wei, Chenfeng Xu, Boyi Li, Masayoshi Tomizuka, Andrea Bajcsy, and Ran Tian. Reimagination with test-time observation interventions: Distractor-robust world model predictions for visual model predictive control. *arXiv preprint arXiv:2506.16565*, 2025.

[4] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[5] Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5152–5161, 2022.

[6] Chuan Guo, Yuxuan Mu, Muhammad Gohar Javed, Sen Wang, and Li Cheng. Momask: Generative masked modeling of 3d human motions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1900–1910, 2024.

[7] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[8] Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. Motiongpt: Human motion as a foreign language. *Advances in Neural Information Processing Systems*, 36:20067–20079, 2023.

[9] Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. Robomonkey: Scaling test-time sampling and verification for vision-language-action models. *arXiv preprint arXiv:2506.17811*, 2025.

[10] Qiayuan Liao, Takara E Truong, Xiaoyu Huang, Yuman Gao, Guy Tevet, Koushil Sreenath, and C Karen Liu. Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion. *arXiv preprint arXiv:2508.08241*, 2025.

[11] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pages 39578–39601, 2024.

[12] Zhengyi Luo, Jinkun Cao, Kris Kitani, Weipeng Xu, et al. Perpetual humanoid control for real-time simulated avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10895–10904, 2023.

[13] Zhengyi Luo, Ye Yuan, Tingwu Wang, Chenran Li, Sirui Chen, Fernando Castaneda, Zi-Ang Cao, Jiefeng Li, David Minor, Qingwei Ben, et al. Sonic: Supersizing motion tracking for natural humanoid whole-body control. *arXiv preprint arXiv:2511.07820*, 2025.

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

[15] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. In *International Conference on Learning Representations*, volume 2024, pages 51772–51783, 2024.

[16] OpenAI. Learning to reason with LLMs. *OpenAI Blog*, 2024. https://openai.com/index/learning-to-reason-with-llms/.

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

[18] Davis Rempe, Mathis Petrovich, Ye Yuan, Haotian Zhang, Xue Bin Peng, Yifeng Jiang, Tingwu Wang, Umar Iqbal, David Minor, Michael de Ruyter, et al. Kimodo: Scaling controllable human motion generation. *arXiv preprint arXiv:2603.15546*, 2026.

[19] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[20] Zelin Tao, Zeran Su, Peiran Liu, Jingkai Sun, Wenqiang Que, Jiahao Ma, Jialin Yu, Jiahang Cao, Pihai Sun, Hao Liang, et al. Heracles: Bridging precise tracking and generative synthesis for general humanoid control. *arXiv preprint arXiv:2603.27756*, 2026.

[21] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H Bermano. Human motion diffusion model. *arXiv preprint arXiv:2209.14916*, 2022.

[22] Wanjiang Weng, Xiaofeng Tan, Junbo Wang, Guo-Sen Xie, Pan Zhou, and Hongsong Wang. Realign: text-to-motion generation via step-aware reward-guided alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 10621–10629, 2026.

[23] Yilin Wu, Ran Tian, Gokul Swamy, and Andrea Bajcsy. From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment. *arXiv preprint arXiv:2502.01828*, 2025.

[24] Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Yong Zhang, Hongwei Zhao, Hongtao Lu, Xi Shen, and Ying Shan. Generating human motion from textual descriptions with discrete representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14730–14740, 2023.

[25] Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *IEEE transactions on pattern analysis and machine intelligence*, 46(6):4115–4128, 2024.

[26] Zewei Zhang, Kehan Wen, Michael Xu, Junzhe He, Chenhao Li, Takahiro Miki, Clemens Schwarke, Chong Zhang, Xue Bin Peng, and Marco Hutter. Learning whole-body humanoid locomotion via motion generation and motion tracking. *arXiv preprint arXiv:2604.17335*, 2026.

## APPENDIX A
## RELATED WORK

**Language-conditioned motion generation.** A prominent line of work represents continuous motion as discrete token sequences. MotionGPT [8] and T2M-GPT [24] first learn a VQ-VAE-style motion tokenizer and then train an autoregressive language-model backbone to generate motion tokens from text. MoMask [6] follows the same discrete-token perspective, but uses an RVQ-VAE to construct hierarchical residual motion tokens and generates them through iterative masked prediction rather than left-to-right autoregressive decoding. In parallel, diffusion-based methods such as MDM [21] and MotionDiffuse [25] formulate text-to-motion generation as an iterative denoising process, providing an alternative paradigm with strong motion diversity, controllability, and flexible conditioning. More recently, Kimodo [18] scales kinematic motion diffusion with a large motion-text dataset and supports text prompting together with rich kinematic constraints such as keyframes, waypoints, joint constraints, and foot contacts. While these works primarily focus on improving the quality and diversity of generated motions, our work addresses the deployment-time trade-off between semantic alignment and downstream tracker compatibility as a plug-in selection module for existing text-conditioned motion generators.

**Aligning motion generation with tracking.** To improve the executability of generated motions, many prior works align motion generation and tracking during training by using the same motion corpus for both the generator and the tracker. This encourages distributional alignment between generated references and the tracker's training data [13, 20]. Some methods further fine-tune the tracker with RL under randomized commands, terrains, or environment configurations while keeping the generator fixed [26]. Such training-time alignment can improve executability, but it couples executability to how the generator and tracker are trained together. In contrast, our work keeps the generator and tracker fixed and introduces controller awareness through a lightweight runtime verifier, which can be adapted without retraining the motion generator.

**Test-time scaling of robot motion generation models.** Large-scale pretraining has been the dominant route to capable generative models, but it is computationally expensive and ties each model's behavior to its training distribution. Recently, test-time scaling, generating multiple candidates and using a verifier to select among them, has emerged as a powerful complement in large language models [11, 19, 16, 7]. Its appeal is that it leverages the diversity already latent in a frozen model to adapt to deployment-time signals (e.g., human preference, task constraints, downstream executability) without retraining the model itself. The same paradigm has recently begun to be applied to vision–language–action (VLA) policies [9, 3, 23], where additional inference-time samples paired with a learned verifier improve manipulation precision; this line of work, however, has so far targeted relatively simple embodiments and arm-only end-effector actions. In this work, we bring the same generate-then-verify paradigm to language-conditioned humanoid whole-body motion generation, where it differs along three axes. The candidates are whole-body high-dimensional motions rather than just end-effector actions, so verification must operate in a much more complex action space. Instead of a reward model trained only to evaluate the semantic or kinematic quality of the generated motion, as in recent text-to-motion test-time alignment methods such as ReAlign [22], whose step-aware reward scores candidate motions on semantic alignment and kinematic realism alone, we make the verifier controller-aware by distilling it from whole-body tracker rollouts. This is critical for humanoid whole-body motion: a kinematically plausible reference can still fall outside the tracker's executable envelope due to balance, contact, or actuation limits, so a motion-only reward cannot certify deployability, only a controller-grounded one can.

## APPENDIX B
## LANGUAGE-CONDITIONED MOTION GENERATOR

This appendix details the FSQ-GPT generator $\mathcal{G}$ used in the TEXT stage (Section III-A): the motion corpus and its preprocessing (Section B-A), the FSQ tokenizer that discretizes motion (Section B-B), and the language model that generates motion tokens from text (Section B-C).

### A. Data Construction

*a) Motion representation:* All motions are expressed in a 36-dimensional Unitree G1 representation per frame: root position $\mathbf{p}_{\text{root}} \in \mathbb{R}^3$, root quaternion (wxyz) $\mathbf{q}_{\text{root}} \in \mathbb{R}^4$, and 29 joint positions $\boldsymbol{\theta} \in \mathbb{R}^{29}$, sampled at $50\,\text{Hz}$.

*b) Corpora and splits:* The generator is trained on a combined corpus of AMASS [14] datasets retargeted to G1 (using HumanML3D captions as the text source) and the CLAW [2] dataset. The merged dataset is split 8:1:1 into train/val/test, and this split is fixed before training any downstream module so that the FSQ tokenizer, the language model, the Dynamic Feasibility Verifier, and the Semantic Alignment Verifier all share the same held-out test motions. Sample counts are reported in Table 4. Each motion is paired with 2-5 language instructions, resulting in a total of 9,116 held-out test prompts.

*c) Module-specific preprocessing:* Each downstream module normalizes the shared 36-dim motion differently: the FSQ tokenizer applies per-channel standardization; the Dynamic Feasibility Verifier augments each frame with first- and second-order finite differences and applies z-score normalization with clipping (Section C-A); and the Semantic Alignment Verifier replaces the root $XY$ position with frame-to-frame velocity to obtain a global-position-invariant embedding (Section C-B).

### B. FSQ Tokenizer

*a) Architecture:* The tokenizer is a Finite Scalar Quantization (FSQ) motion VAE [15]. A $50\,\text{Hz}$, 36-dim G1 motion is encoded by two stride-2 temporal layers (encoder width 512, depth 3, LayerNorm, ReLU) into a code sequence of

TABLE 4: Sample counts of the combined motion corpus after preprocessing.

| **Corpus** | **Train** | **Val** | **Test** | **Total** |
|---|---|---|---|---|
| AMASS (with HumanML3D text) | 14148 | 1768 | 1768 | 17684 |
| CLAW | 11977 | 1497 | 1497 | 14971 |
| Combined | 26125 | 3265 | 3265 | 32655 |

TABLE 5: FSQ reconstruction loss weights $\lambda_c$.

| root pos | root quat | joint pos | root vel | joint vel | root acc | joint acc |
|---|---|---|---|---|---|---|
| 1.0 | 1.0 | 2.0 | 1.0 | 1.0 | 0.5 | 0.5 |

length $L = T/4$. The FSQ quantizer uses the level vector $[3, 3, 3, 3, 3, 2, 2, 2, 2, 2]$, yielding a fixed codebook of $K = \prod_i \ell_i = 7{,}776$ entries. We adopt FSQ over a learned VQ-VAE codebook to avoid codebook collapse on our comparatively small humanoid corpus, which we found important for stable training.

*b) Reconstruction loss:* Inputs are per-channel standardized and supervised with a SmoothL1 reconstruction loss on the pose together with its first- and second-order temporal derivatives:

$$\mathcal{L}_{\text{fsq}} = \sum_{c \in \mathcal{C}} \lambda_c \, \text{SmoothL1}(\mathbf{m}_c, \hat{\mathbf{m}}_c) \,, \tag{6}$$

with per-term weights $\lambda_c$ given in Table 5. Joint positions are weighted most heavily because they directly drive the downstream tracker.

*c) Training:* Optimization and schedule hyperparameters are summarized in Table 6; the tokenizer is trained over sliding windows of 100 frames.

### C. Language Model

*a) Tokenized sequence-to-sequence interface:* We cast text-to-motion as a sequence-to-sequence problem over a shared vocabulary. The frozen FSQ tokenizer maps a continuous motion $\mathbf{m} \in \mathbb{R}^{T \times 36}$ to a discrete code sequence $\mathbf{z} = (z_1, \ldots, z_L)$ with $L = T/4$ and $z_\ell \in \{1, \ldots, K\}$, $K = 7{,}776$. We extend the Flan-T5-base [4] vocabulary with these $K$ codes together with three reserved special tokens marking the start, end, and padding of a motion sequence; the end-of-motion token lets the decoder determine the generated motion length at inference time. Text is truncated to 50 sub-word tokens and motion to $[16, 2048]$ frames, i.e. $[4, 512]$ motion tokens after the tokenizer's $4\times$ temporal downsampling.

*b) Architecture:* We initialize from an instruction-tuned Flan-T5 ($\sim$220M parameters), whose encoder provides a strong language-understanding prior for conditioning on diverse instructions, while the decoder weights serve as a warm start for autoregressive generation over the extended motion-token vocabulary.

*c) Training objective:* The model is fine-tuned on paired $(\ell, \mathbf{m})$ data with teacher forcing under the standard autoregressive cross-entropy over motion tokens,

$$\mathcal{L}_{\text{lm}} = -\sum_{\ell=1}^{L} \log p_\theta(z_\ell \mid z_{<\ell}, \, \text{enc}(\ell)) \,, \tag{7}$$

where $\text{enc}(\ell)$ denotes the T5 encoder states for the instruction.

*d) Sampling for candidate generation:* At inference, the $N$ candidates of the TEXT stage (Section III-A) are drawn i.i.d. by ancestral sampling, $\mathbf{z}_i \sim p_\theta(\cdot \mid \ell)$, and each token sequence is decoded back to a 36-dim trajectory by the frozen FSQ decoder. The diversity of the candidate pool is controlled by the sampling temperature $\tau$ together with top-$k$ / nucleus (top-$p$) truncation: a larger $\tau$ widens the feasible and semantic spread that the verifiers select from, at the cost of more low-quality samples. All training and sampling hyperparameters are summarized in Table 6.

## APPENDIX C
## VERIFIER TRAINING

### A. Dynamic Feasibility Verifier

*a) SONIC roll-outs and oracle labels:* Oracle feasibility labels are collected by rolling out every reference motion offline through the SONIC whole-body tracker [13] in MuJoCo. Each episode is initialized from the first reference frame and advances until the clip ends or an early-termination condition fires. We adopt the height-based termination criteria of BeyondMimic [10]; to avoid duplication, the precise thresholds, the success flag $y_s$, and the progress ratio $q_g$ are defined once in Section E and reused verbatim here. Each roll-out thus yields the three signals consumed by Equation (1): the binary success flag $y_s$, the progress ratio $q_g$, and a normalized tracking quality

$$q_d(\mathbf{m}) = \tfrac{1}{2}\Big[\text{clip}\Big(1 - \tfrac{e_{\text{acc}}}{e^{95}_{\text{acc}}}, 0, 1\Big) + \text{clip}\Big(1 - \tfrac{e_{\text{vel}}}{e^{95}_{\text{vel}}}, 0, 1\Big)\Big] \,, \tag{8}$$

where $e_{\text{acc}}$ and $e_{\text{vel}}$ are the acceleration and velocity tracking errors and $e^{95}$ denotes their 95th-percentile normalizers.

*b) Ordering guarantee:* The constraint $\beta < 1/(1+\alpha)$ makes $Q^*$ *feasibility-first*: any successful roll-out scores above any failed one, regardless of other metrics.

**Proposition 1** (Feasibility-first ordering)**.** *For $\alpha, \beta > 0$, $y_s \in \{0, 1\}$ and $q_d, q_g \in [0, 1]$, the quality $Q^*$ of Equation* (1)

TABLE 6: Training and sampling configuration of the FSQ-GPT generator.

| | **FSQ Tokenizer** | **Language Model** |
|---|---|---|
| Backbone / init | temporal-conv VAE | Flan-T5-base (instr.-tuned) |
| Optimizer | AdamW | AdamW |
| Learning rate | $4 \times 10^{-5}$ | $8 \times 10^{-4}$ |
| $(\beta_1, \beta_2)$ | $(0.9, 0.99)$ | $(0.9, 0.99)$ |
| Weight decay | 0 | 0 |
| LR schedule | cosine | cosine |
| Batch size | 256 | 32 |
| Epochs | 100 | early-stop (val.) |
| Sampling | — | temp. 1.0, top-$k$=50, top-$p$=0.95 |

TABLE 7: Prediction accuracy of the three Dynamic Verifier heads on the held-out test set.

| **Head** | **Metric** | **Value** |
|---|---|---|
| $\hat{p}_s$ (success) | AUROC↑ | 0.979 |
| | AUPRC↑ | 0.997 |
| | Fail recall↑ | 0.930 |
| $\hat{q}_d$ (dynamics) | Spearman $\rho$ vs. $E_{\text{acc}}$↓ | $-0.936$ |
| | Spearman $\rho$ vs. $E_{\text{vel}}$↓ | $-0.932$ |
| $\hat{q}_g$ (progress) | Spearman $\rho$ vs. progress ratio↑ | 0.238 |
| | Spearman $\rho$ vs. progress (on failures)↑ | 0.925 |

*satisfies* $Q^*(1, \cdot, \cdot) > Q^*(0, \cdot, \cdot)$ *for all arguments if and only if* $\beta < \frac{1}{1+\alpha}$.

*Proof:* With $q_d, q_g \in [0, 1]$, the two regimes of Equation (1) are bounded as

$$Q^*(1, q_d, q_g) = \frac{1 + \alpha q_d}{1 + \alpha} \in \left[\tfrac{1}{1+\alpha},\, 1\right],$$
$$Q^*(0, q_d, q_g) = \beta\, q_g\, q_d \in [0,\, \beta],$$

and both lower/upper endpoints are attained ($q_d = 0$ and $q_d = q_g = 1$, respectively). The successful range therefore lies strictly above the failed range iff its lower end exceeds the failed upper end, i.e. $\frac{1}{1+\alpha} > \beta$. ■

The same bounds show $Q^*$ stays graded *within* each group: it increases in $q_d$ among successes and in $q_g q_d$ among failures, so the "least-bad" candidate is still preferred when all candidates fail (Table 9).

TABLE 8: $R_{\text{dyn}}$ **ranks candidates in close agreement with the simulator oracle without ever running it.** Within-prompt Kendall $\tau$ between $R_{\text{dyn}}$ and the oracle quality $Q^*$ on 9,116 held-out prompts.

| **Prompt type** | **#Prompts** | **Kendall $\tau$ ↑** |
|---|---|---|
| Mixed (success + failure) | 6,759 | 0.674 |
| All-failure | 46 | 0.357 |
| All-success | 2,311 | 0.608 |
| Overall | 9,116 | **0.656** |

*c) Model structure and training strategy:* Each raw 36-dim frame is expanded into a 94-dim feature (root dynamics 7, joint positions 29, velocities 29, accelerations 29), z-score normalized and clipped to $[-10, 10]$. An input projection encodes each semantic group to 128-dim and fuses them into a 256-dim token; a 4-layer causal Transformer ($d_{\text{model}} = 256$, 4 heads, pre-LayerNorm) followed by mean-attention pooling and three lightweight MLP heads outputs $(\hat{p}_s, \hat{q}_d, \hat{q}_g)$, which are recombined through Equation (2) with $\alpha = 0.4,\ \beta = 0.6$. The three heads are trained jointly by minimizing, over each batch $\mathcal{B}$,

$$\mathcal{L} = \underbrace{\mathrm{BCE}_{w^+}\big(\hat{p}_s,\, y_s\big)}_{\text{success}} + \lambda_d \underbrace{\big(\hat{q}_d - q_d\big)^2}_{\text{dynamics}} + \lambda_g \underbrace{\frac{\sum_{i\in\mathcal{B}}(1 - y_s^i)\big(\hat{q}_g^i - q_g^i\big)^2}{\sum_{i\in\mathcal{B}}(1 - y_s^i)}}_{\substack{\text{progress} \\ \text{(failed roll-outs only)}}}, \tag{9}$$

where $\lambda_d = 0.6$, $\lambda_g = 0.8$, $\mathrm{BCE}_{w^+}$ is a binary cross-entropy with positive-class weight $w^+$ to offset class imbalance, and the first two terms are averaged over $\mathcal{B}$. The factor $(1 - y_s)$ masks the progress loss to *failed* roll-outs: a success completes the reference, so $q_g \equiv 1$ is a constant copy of $y_s$ that carries no gradient, whereas on failures $q_g$ varies and is exactly where accurate progress estimates are needed for partial-credit ranking, hence the head is weak on progress overall but strong on failures (Table 7).

*d) Fidelity studies:* We validate $R_{\text{dyn}}$ on held-out data along two axes: per-head prediction accuracy (Table 7) and

TABLE 9: Reward-design ablation at $N$=8. Model weights are fixed; only the inference-time reward formula varies. †mean progress on all-failure prompts.

| | Reward formula | Succ↑ | $\bar{Q}^*$ ↑ | All-fail prog.† ↑ |
|---|---|---|---|---|
| (a) | $\hat{p}_s$ only | **0.984** | 0.922 | 0.395 |
| (b) | $\hat{q}_d$ only | 0.953 | 0.916 | 0.416 |
| (c) | $\hat{q}_g$ only | 0.982 | 0.921 | **0.427** |
| (d) | $\hat{p}_s \cdot \hat{q}_d$ | 0.981 | **0.931** | 0.411 |
| (e) | $\hat{p}_s \cdot \hat{q}_g$ | **0.984** | 0.922 | 0.395 |
| (f) | $Q^*$ formula, Equation (2) (ours) | <u>0.983</u> | **0.931** | <u>0.425</u> |

end-to-end ranking agreement with the oracle (Table 8). The success head $\hat{p}_s$ is highly discriminative (AUROC 0.979, AUPRC 0.997) and maintains 0.930 recall on the minority failure class, so infeasible candidates are rarely mislabeled as feasible. The dynamics head $\hat{q}_d$ is strongly rank-correlated with both tracking-error signals (Spearman $\rho = -0.936$ and $-0.932$ against acceleration and velocity errors, negative by construction). The progress head $\hat{q}_g$ has a weak global correlation with the progress ratio ($\rho = 0.238$) because most successful roll-outs cluster near $q_g = 1$, but becomes highly informative on the failed roll-outs where progress genuinely varies ($\rho = 0.925$), precisely the regime where partial-credit ranking is needed. At the composite level, the within-prompt Kendall $\tau$ between $R_{\text{dyn}}$ and $Q^*$ over 9,116 held-out prompts is 0.656 overall, rising to 0.674 on *mixed* prompts containing both feasible and infeasible candidates (Table 8), confirming that $R_{\text{dyn}}$ recovers the bulk of the oracle ordering without invoking the tracker.

*e) Ablation studies:* We ablate the two design choices that turn the three heads into a single selection reward: the functional *form* of the recombination, and the *weights* it places on each term. Table 9 varies the inference-time reward formula at $N$=8 with the model weights held fixed. Using $\hat{p}_s$ alone yields the highest raw success rate but collapses to near-random ordering once every candidate in a prompt fails, because it provides no signal to separate equally-infeasible motions; conversely, $\hat{q}_d$ alone breaks the success-first hierarchy and lets smooth-but-failing candidates win. The full $Q^*$ formula of Equation (2) strikes the best balance: it matches the strongest variants on global quality ($\bar{Q}^* = 0.931$) while retaining a high all-failure progress score (0.425), so it still selects the "least bad" candidate when no motion succeeds.

### B. Semantic Alignment Verifier

*a) Model structure and training strategy:* The semantic verifier follows the T2M co-embedding design [5] but is trained *directly* on the G1 skeleton so that its embedding space is consistent with the candidates produced by the TEXT stage. The motion encoder applies two temporal Conv1d downsampling layers (a MovementConvEncoder) followed by a BiGRU to produce a 512-dim motion embedding $\varphi_{\text{motion}}(\mathbf{m})$; the text encoder sums 300-dim word and 15-dim part-of-speech embeddings and feeds them through a BiGRU to obtain $\varphi_{\text{text}}(\ell)$. The ConvEncoder is first pretrained as a motion autoencoder and frozen, after which the two BiGRU encoders are trained jointly with the all-pairs margin contrastive objective in Equation (3) ($\delta = 2.0$). The test-time score $R_{\text{text}}$ is the exponentiated negative embedding distance.

*b) Fidelity studies:* We verify that the contrastive embedding behind $R_{\text{text}}$ has actually learned a separating geometry. On the 9,116 held-out pairs and a 32-distractor pool, $R_{\text{text}}$ attains R@1 = 0.747 and R@3 = 0.935 in motion-to-text retrieval, far above the 1/32 random baseline (Table 10); shuffling the motion input collapses retrieval to near-random, confirming that the score captures text–motion correspondence rather than a marginal motion prior.

TABLE 10: $R_{\text{text}}$ **separates paired text and motion.** Retrieval sanity check on the held-out test set with a 32-distractor pool. Shuffling the motion input collapses every metric to near-random.

| Metric | Paired (test) | Shuffled (control) |
|---|---|---|
| R@1↑ | **0.747** | 0.014 |
| R@2↑ | **0.885** | 0.059 |
| R@3↑ | **0.935** | 0.102 |
| Matching score↓ | **0.248** | 2.977 |
| Gap (neg−pos)↑ | **2.835** | 0.019 |

## APPENDIX D
## REAL-WORLD RESULTS

*a) Hardware setup and motion collection:* All real-world experiments are conducted on a Unitree G1 humanoid robot using the SONIC whole-body tracking policy [13]. We evaluate 30 text prompts spanning locomotion, upper-body gestures, and their combinations. These motions are used to assess whether the references selected by **TEXEDO** can be executed reliably on physical hardware.

*b) Results:* **TEXEDO** successfully executes all 30 real-world trajectories on the physical robot. Figure 6 in the main text shows representative execution sequences, and Table 11 reports the per-prompt real-world tracking metrics.

TABLE 11: Per-prompt real-world deployment metrics. A checkmark indicates that the robot completes the trajectory without falling or early termination. Tracking errors are reported in mm, mm/frame, and mm/frame$^2$, respectively; the overall row reports frame-weighted averages.

| **Prompt** | **Success** | $E_{\text{mpjpe-l}} \downarrow$ | $E_{\text{vel}} \downarrow$ | $E_{\text{accel}} \downarrow$ |
|---|---|---|---|---|
| the man is doing salsa dance | ✓ | 39.14 | 5.60 | 3.60 |
| the person took a big step over something. | ✓ | 26.18 | 2.84 | 1.90 |
| a person walks in a circle clockwise. | ✓ | 23.11 | 3.13 | 2.42 |
| A person appears happy while throwing punches and squatting forward. | ✓ | 47.38 | 4.68 | 2.59 |
| a person walks forward slowly. | ✓ | 20.04 | 1.97 | 1.35 |
| The person holds an object and moves forward. | ✓ | 29.35 | 3.43 | 1.72 |
| A person is carrying something heavy while moving forward. | ✓ | 42.22 | 3.64 | 2.21 |
| A person throws a right hook ahead. | ✓ | 40.76 | 5.46 | 3.08 |
| A person dances happily ahead. | ✓ | 32.18 | 4.90 | 2.83 |
| a person walks straight forward | ✓ | 17.93 | 2.12 | 1.55 |
| A person executes a left jab forward with force and then throws a left hook ahead with full force. | ✓ | 46.58 | 6.27 | 3.40 |
| A person moves cautiously right. | ✓ | 21.70 | 2.33 | 1.44 |
| The person moves while boxing and walking forward. | ✓ | 32.67 | 4.26 | 2.51 |
| A person jabs right forward and performs a happy dance. | ✓ | 41.41 | 4.30 | 2.50 |
| A person advances while boxing ahead aggressively. | ✓ | 38.68 | 4.31 | 2.20 |
| the person is stretching, with their arms above their head. | ✓ | 34.59 | 0.90 | 0.66 |
| A person carries an object sideways to the right balancing carefully. | ✓ | 43.68 | 2.99 | 1.93 |
| A person throws a left hook ahead with full force. | ✓ | 41.04 | 6.70 | 4.00 |
| spinning in a circle in the air. | ✓ | 48.91 | 11.41 | 11.09 |
| A person shambles left lifelessly. | ✓ | 41.55 | 4.66 | 2.60 |
| a man is doing jumping jacks. | ✓ | 33.68 | 6.12 | 4.15 |
| a person walking forward aggressively | ✓ | 21.24 | 2.37 | 1.92 |
| A person dances joyfully forward with enthusiasm. | ✓ | 30.10 | 4.76 | 2.75 |
| A person pivots right while moving cheerfully. | ✓ | 49.25 | 5.98 | 3.38 |
| A person rotates right while sneaking. | ✓ | 33.77 | 4.77 | 2.56 |
| A person moves while crouching rearward low to the ground. | ✓ | 58.24 | 4.49 | 2.08 |
| A person bounces along sideways to the left cheerfully and then scurries away rearward frantically. | ✓ | 39.62 | 2.75 | 1.71 |
| A person scurries away sideways to the left in a panic. | ✓ | 36.96 | 3.83 | 2.32 |
| a man raises his right hand to his head and then returns it to his side | ✓ | 23.47 | 1.82 | 1.43 |
| The person stumbles backward in a slow, zombie-like shamble. | ✓ | 36.30 | 3.36 | 2.09 |
| **Overall** | **30/30** | **37.41** | **4.35** | **2.67** |

## APPENDIX E
## METRICS

We report two families of metrics. Physical metrics measure how well the deployment-time tracker executes the selected motion; the semantic metric measures whether the motion realizes the text instruction.

*a) Notation:* A roll-out tracks a reference of $T$ frames over $J$ rigid bodies. At frame $t$ let $\mathbf{p}_t^{\text{ref},j}, \mathbf{p}_t^{\text{rob},j} \in \mathbb{R}^3$ be the world position of body $j$ in the reference and in the realized robot state, and let $\mathbf{q}_t^{\text{ref},0}, \mathbf{q}_t^{\text{rob},0}$ be the orientation of the anchor body (pelvis, $j$=0). We write $[\cdot]_z$ for the vertical component and $\mathbf{R}(\mathbf{q}) \in \mathrm{SO}(3)$ for the rotation matrix of a quaternion.

*b) Termination criteria:* Following BeyondMimic [10], the episode is terminated at the first frame $t$ at which any of the following *height-based* conditions fires:

$$\text{(anchor height)} \quad \left|[\mathbf{p}_t^{\text{ref},0}]_z - [\mathbf{p}_t^{\text{rob},0}]_z\right| > h_a, \tag{10}$$

$$\text{(anchor tilt)} \quad \left|\left[\mathbf{R}(\mathbf{q}_t^{\text{ref},0})^\top \mathbf{g}\right]_z - \left[\mathbf{R}(\mathbf{q}_t^{\text{rob},0})^\top \mathbf{g}\right]_z\right| > c_o, \tag{11}$$

$$\text{(end-effector height)} \quad \max_{b \in \mathcal{B}_{\text{ee}}} \left|[\tilde{\mathbf{p}}_t^{\text{ref},b}]_z - [\tilde{\mathbf{p}}_t^{\text{rob},b}]_z\right| > h_e, \tag{12}$$

where $\mathbf{g} = (0, 0, -1)^\top$ is the gravity direction, $\mathcal{B}_{\text{ee}}$ is the set of end-effector bodies, $\tilde{\mathbf{p}}$ denotes anchor-relative body positions, and $(h_a, c_o, h_e) = (0.25, 0.8, 0.25)$. Let $\tau$ be the first frame at which Equations (10) to (12) fire, or $\tau = T$ if the roll-out reaches the end of the reference. Equation (11) compares the $z$-component of gravity projected into the reference and robot anchor frames, i.e. the deviation in anchor tilt.

*c) Success and progress:*

$$\text{Succ} = \mathbb{1}[\tau = T], \qquad q_g = \frac{\tau}{T} \in (0, 1], \tag{13}$$

so a roll-out is **successful** (↑) only if it reaches the final reference frame without any termination, and the progress ratio $q_g$ is the fraction of the reference completed before termination.

*d) Tracking errors:* With the position finite differences $\mathbf{v}_t^{\cdot,j} = \mathbf{p}_t^{\cdot,j} - \mathbf{p}_{t-1}^{\cdot,j}$ and $\mathbf{a}_t^{\cdot,j} = \mathbf{v}_t^{\cdot,j} - \mathbf{v}_{t-1}^{\cdot,j}$, and the anchor-removed positions $\tilde{\mathbf{p}}_t^{\cdot,j} = \mathbf{p}_t^{\cdot,j} - \mathbf{p}_t^{\cdot,0}$, the tracking errors (in

mm, mm/frame, and mm/frame$^2$) are

$$E_{\text{mpjpe-l}} = \frac{10^3}{T}\sum_{t=1}^{T}\frac{1}{J}\sum_{j=1}^{J}\left\|\tilde{\mathbf{p}}_t^{\text{ref},j} - \tilde{\mathbf{p}}_t^{\text{rob},j}\right\|_2, \tag{14}$$

$$E_{\text{vel}} = \frac{10^3}{T-1}\sum_{t=2}^{T}\frac{1}{J}\sum_{j=1}^{J}\left\|\mathbf{v}_t^{\text{ref},j} - \mathbf{v}_t^{\text{rob},j}\right\|_2, \tag{15}$$

$$E_{\text{accel}} = \frac{10^3}{T-2}\sum_{t=3}^{T}\frac{1}{J}\sum_{j=1}^{J}\left\|\mathbf{a}_t^{\text{ref},j} - \mathbf{a}_t^{\text{rob},j}\right\|_2. \tag{16}$$

$E_{\text{mpjpe-l}}$ (↓) is the root-relative (pelvis-anchored) mean per-body position error; $E_{\text{vel}}$ (↓) and $E_{\text{accel}}$ (↓) penalize first- and second-order kinematic mismatch and the factor $10^3$ converts m to mm. As in the main experiments, $E_{\text{mpjpe-l}}$, $E_{\text{vel}}$, and $E_{\text{accel}}$ are averaged over the executed frames of *successful* rollouts only, so that tracking error is not confounded with early termination. Finally, $Q^*$ (↑, Equation (1)) collapses success, tracking quality, and progress into the single oracle quality used as the Dynamic Verifier supervision target.

*e) Semantic metric:* To obtain an independent, scalable measure of how well a rendered rollout realizes its text prompt, we adopt a VLM-as-judge ensemble that scores each (text, rendered-rollout video) pair with several low-temperature VLM calls and reports their mean as a single score in $[1, 10]$. The ensemble varies along two main axes to ensure robustness. First, to mitigate the risk of the metric inheriting the idiosyncratic biases of any single model, we query two distinct VLMs from different families, namely GPT-5.5 and `gemini-2.5-flash`, and average their outputs. Second, each model evaluates the video under a $2 \times 2$ grid combining two evaluation rubrics and two frame counts. The first rubric is a holistic rubric, which tags five key semantic units (ACTION, BODY-PART, SPATIAL, TEMPORAL, ATTRIBUTE) as MATCH, PARTIAL, or MISMATCH, mapping the total fraction of successful matches to a 1–10 scale. The second is a per-axis rubric, which scores the same five dimensions individually on a $\{0, 1, 2\}$ scale and sums them. Both rubrics are evaluated at both 8 and 16 uniformly-sampled frames. The denser 16-frame sampling is crucial for catching temporal failures, such as incorrect repetition counts, improper pacing, or trajectory reversals, that might be aliased or missed at 8 frames. Additionally, both rubrics incorporate an agent-equivalence clause, which dictates that the humanoid robot stands in for any human referent in the prompt, as well as strict floor caps. These floor caps ensure that a single critical failure, such as an incorrect primary action or an opposite-direction translation, cannot be masked or inflated by matching minor filler details. The final reported semantic score is the mean across all eight configuration calls (2 models $\times$ 2 rubrics $\times$ 2 frame counts). The full prompts and rubrics are detailed in Section F.

## APPENDIX F
## VLM-JUDGE PROMPTS

This appendix reproduces, verbatim, the two VLM-judge rubrics used by the ensemble of the previous section. Each rubric is issued to both models (GPT-5.5 and `gemini-2.5-flash`) at two temporal resolutions (8 and 16 uniformly-sampled frames) and at temperature 0, giving the eight calls whose mean is the reported semantic score.

### A. Holistic Rubric

```
# AGENT EQUIVALENCE -- read first
The video shows a humanoid robot performing a motion. For
all evaluation purposes, treat the robot as a stand-in
for any human referent in the text ("a person", "the
man", "they", "this person"). Robot identity is NEVER a
reason to label a BODY unit as MISMATCH; only assess
whether the referenced body part is correctly involved.

# YOUR ROLE
You are a strict evaluator for text-conditioned robot
motion generation. Your job is to find failure modes --
do not be charitable to the generator. Judge only on
observable evidence in the video. If evidence is missing
or ambiguous, do NOT default to MATCH.

# INPUT
- Text: a natural language description of the target
motion.
- Video: either a native mp4, or N uniformly-sampled
frames in temporal order. **Treat the gaps between
samples as continuous motion you did not directly
observe.** A momentary airborne phase between adjacent
frames is normal for walking and is NOT evidence against
"walking".

# THREE-STEP PROTOCOL

## Step 1 -- Semantic Decomposition

Parse the text into atomic semantic units. **Compound
verbs MUST be decomposed into component primitives.**
Examples:
- "crouch-walks forward" -> ACTION 'crouch posture' +
ACTION 'walking gait' + SPATIAL 'forward'
- "throws a right hook" -> ACTION 'throw' + BODY 'right
arm' + ACTION 'hook (curved trajectory)'
- "steps backwards" -> ACTION 'step' + SPATIAL 'backward'
- "carries an object back and then strides sideways to
the right" -> ACTION 'carry object' + SPATIAL 'back' +
TEMPORAL 'then' + ACTION 'stride' + SPATIAL 'sideways
right'

Categories (each unit MUST get exactly one):
- ACTION: a verb or movement primitive ("walk", "throw",
"wave", "jab", "swim")
- BODY: a body part or end-effector ("right arm", "both
hands"). Skip BODY units that just restate the agent --
do NOT create BODY='a person' / BODY='the man'.
- SPATIAL: a direction, trajectory, or target ("forward",
"sideways right", "above head", "in front of them")
- TEMPORAL: sequencing or duration ("first ... then",
"after a moment", "several times", "quickly", "then
stops")
- ATTRIBUTE: stylistic quality, magnitude, or speed
("happily", "powerfully", "slightly", "joyfully"). If a
word describes how a unit is executed rather than what is
executed, it's an ATTRIBUTE.

Do NOT create empty filler units (no ACTION='doing', no
BODY='person'). If a piece of text adds no new
constraint, skip it. Aim for 2-6 units for simple
prompts, up to 8 for compound prompts. Mark the FIRST
ACTION unit you extract as the **primary ACTION** -- it's
the one whose failure caps the score.

## Step 2 -- Per-Unit Verification
```

```
For each unit assign one label. Cite frame numbers (or
timestamps if a native video) as evidence.
- MATCH: clearly satisfied with concrete visible
evidence.
- PARTIAL: roughly satisfied but with a noticeable
deviation. Examples: text says "quickly" but motion is
normal speed; text says "small steps" but steps look
normal-sized; text says "forward" and net translation is
forward but minimal.
- MISMATCH: absent, contradicted, or violated. **If the
unit's realization cannot be verified from the video,
label MISMATCH -- absence of evidence is NOT evidence of
MATCH.**

Be strict. If you find yourself hedging ("seems to
roughly resemble..."), the label is at best PARTIAL.

## Step 3 -- Holistic Scoring (DETERMINISTIC)

The final score is derived from Step 2 labels -- do not
pick a "vibe" number.

**a) Floor rules (apply first, take the lowest applicable
cap):**
- If the video shows no motion related to the text at all
-> score = 1.
- If the primary ACTION unit is MISMATCH -> score <= 3.
- If a SPATIAL unit is MISMATCH because the motion is in
the opposite direction (e.g., text says "forward" but
motion is backward) -> score <= 3.
- If >= 50% of units are MISMATCH -> score <= 3.
- If any unit is MISMATCH and any other unit is MISMATCH
or PARTIAL -> score <= 7.
- If any unit is PARTIAL (and no MISMATCH) -> score <= 8.

**b) Candidate score from label fraction:**
Let M = number of MATCH, P = number of PARTIAL, X =
number of MISMATCH, N = M+P+X.
Compute f = (M + 0.5*P) / N. Then:
- f >= 0.95 -> candidate in {9, 10}
- 0.75 <= f < 0.95 -> candidate in {7, 8}
- 0.50 <= f < 0.75 -> candidate in {5, 6}
- 0.25 <= f < 0.50 -> candidate in {3, 4}
- f < 0.25 -> candidate in {1, 2}

Inside each band, choose the higher integer if the
matched units include the primary ACTION and no deviation
feels severe; otherwise the lower one.

**c) Final score = min(floor_cap, candidate).**

You may freely assign 2, 4, 7, 8, 9 -- they are valid
scores. Do not collapse to round numbers like 1/3/5/10.

# OUTPUT -- JSON only, no markdown fences, no commentary
before or after

Output a single valid JSON object with exactly the fields
shown below. Use real integers and real strings -- do NOT
include angle-bracket placeholders like `<int>` in your
output.

Worked example for a hypothetical prompt "walk forward
slowly" where the robot walks forward but at normal
speed:
{
  "semantic_units": [
    {"id": 1, "category": "ACTION", "content": "walk",
"label": "MATCH", "rationale": "Robot performs a bipedal
walking gait with alternating steps from frame 2 through
frame 8."},
    {"id": 2, "category": "SPATIAL", "content":
"forward", "label": "MATCH", "rationale": "Robot
translates forward across the floor pattern; visible
position change between frame 1 and frame 8."},
    {"id": 3, "category": "ATTRIBUTE", "content":
"slowly", "label": "PARTIAL", "rationale": "Pace looks
like normal walking, not slow."}
  ],
  "score": 7,
  "justification": "Primary ACTION 'walk' MATCH and
SPATIAL MATCH, but ATTRIBUTE 'slowly' is PARTIAL. f = (2
+ 0.5*1)/3 = 0.83, band {7,8}. Floor cap from one
PARTIAL: <= 8. Final: 7, downgraded inside the band
because the speed deviation is not minor."
}

Produce exactly that JSON shape (same keys, same
nesting). Each unit id is an integer starting at 1 and
incrementing. Score is an integer 1..10. No other
top-level keys.
```

### B. Per-Axis Rubric

```
# AGENT EQUIVALENCE
The video shows a humanoid robot. Treat the robot as a
stand-in for any "person" / "man" / "they" referenced in
the text. Robot identity is never a reason to penalize.

# YOUR ROLE
You are a strict evaluator for text-conditioned robot
motion. Judge only on visible evidence in the frames.
Treat unsampled gaps between frames as continuous motion
you didn't observe -- do NOT use those gaps as evidence
of failure.

# INPUT
- Text: a description of the target motion.
- Video: a native mp4, or N uniformly-sampled frames in
temporal order.

# FIVE-AXIS PROTOCOL

For each axis below, assign an integer score in {0, 1,
2}:

**Axis A -- PRIMARY ACTION**: the main verb of the text
(walk, jab, wave, swim, crawl, dance, ...).
  - 2 = robot clearly performs the primary action.
  - 1 = robot does something resembling the primary
action with significant deviation, OR performs an action
that overlaps partially.
  - 0 = robot does something unrelated, or no discernible
motion.

**Axis B -- BODY PART CORRECTNESS**: did the correct body
part(s) execute the action? (e.g., "right arm" jabbed,
"both hands" waved.)
  - 2 = exactly the body parts named in the text.
  - 1 = wrong-side limb, OR additional body parts also
involved beyond what was specified.
  - 0 = action executed by a completely wrong body part,
OR the named body part is idle while another acts.
  - If the text does not specify a body part, score 2 by
default.

**Axis C -- SPATIAL CORRECTNESS**: direction, trajectory,
target ("forward", "sideways right", "in front of them",
"back and then to the side").
  - 2 = all spatial constraints satisfied.
  - 1 = some satisfied, some violated; OR direction
approximately right but small/limited.
  - 0 = opposite or unrelated direction. If the text says
"forward" and motion is backward, this is 0.
  - If the text has no spatial component, score 2 by
default.

**Axis D -- TEMPORAL / COUNT / SEQUENCING**: ordering
("first ... then"), duration ("after a moment"),
repetition count ("several times"), pace ("quickly").
```

```
  - 2 = repetition count and sequencing match clearly
(cite frame indices showing each repetition).
  - 1 = motion type right but count / duration / order
partially off (e.g., text says "several times" but motion
only repeats once or twice; or text says "then stops" but
motion continues).
  - 0 = no temporal/count component satisfied at all.
  - If the text has no temporal/count component, score 2
by default.

**Axis E -- ATTRIBUTE / STYLE**: speed, magnitude,
intensity, mood ("slowly", "powerfully", "happily", "with
force", "slightly").
  - 2 = the named quality is clearly observable.
  - 1 = partially evident (e.g., "powerfully" -- motion
is fast but not visibly forceful; "joyfully" -- motion is
dynamic but no clear celebratory quality).
  - 0 = the named quality is absent or contradicted.
  - If the text has no attribute, score 2 by default.

# SCORE MAPPING (DETERMINISTIC)
Let S = A + B + C + D + E (sum, range 0..10).

Final score = max(1, S). However, apply these floor caps
first:
- If A = 0 -> final score <= 3.
- If C = 0 AND text has a spatial component -> final
score <= 3.
- If A <= 1 -> final score <= 7.

# OUTPUT -- JSON only

Worked example for text "A person waves both hands at
their side several times.":
{
  "axes": {
    "A_primary_action": {"score": 2, "rationale": "Robot
raises both hands and oscillates them between frames 2-8,
consistent with waving."},
    "B_body_part":      {"score": 2, "rationale": "Both
hands are clearly involved (text required 'both
hands')."},
    "C_spatial":        {"score": 1, "rationale": "Hands
move at chest/face level rather than 'at their side';
minor spatial deviation."},
    "D_temporal":       {"score": 1, "rationale": "Only
~2 visible waves in the clip; text says 'several times'
which implies more than 2."},
    "E_attribute":      {"score": 2, "rationale": "No
attribute in text -- default 2."}
  },
  "sum": 8,
  "floor_cap_applied": "none",
  "score": 8,
  "justification": "Action and body parts correct; minor
spatial and count deviations bring this to 8."
}

Produce a JSON object with exactly the same shape. Use
real integers 0/1/2 for axis scores. Score is integer
1..10. No markdown fences, no commentary.
```