Bridging Environments and Language with Rendering Functions and Vision-Language Models

Théo Cachet [1 2] **Christopher R. Dance** [1] **Olivier Sigaud** [2]

Abstract

Vision-language models (VLMs) have tremendous potential for *grounding* language, and thus enabling *language-conditioned agents (LCAs)* to perform diverse tasks specified with text. This has motivated the study of LCAs based on reinforcement learning (RL) with rewards given by rendering images of an environment and evaluating those images with VLMs. If single-task RL is employed, such approaches are limited by the cost and time required to train a policy for each new task. Multi-task RL (MTRL) is a natural alternative, but requires a carefully designed corpus of training tasks and does not always generalize reliably to new tasks. Therefore, this paper introduces a novel decomposition of the problem of building an LCA: first find an environment *configuration* that has a high VLM score for text describing a task; then use a (pretrained) goal-conditioned policy to reach that configuration. We also explore several enhancements to the speed and quality of VLM-based LCAs, notably, the use of distilled models, and the evaluation of configurations from multiple viewpoints to resolve the ambiguities inherent in a single 2D view. We demonstrate our approach on the Humanoid environment, showing that it results in LCAs that outperform MTRL baselines in zero-shot generalization, without requiring any textual task descriptions or other forms of environment-specific annotation during training.

1. Introduction

The widespread adoption of large language models (LLMs) and text-to-image models in modern society demonstrates the convenience of natural language interaction with AI models (Bubeck et al., 2023; Rombach et al., 2022). This motivates the study of *language-conditioned agents (LCAs)* that can execute diverse commands, specified with text, in a given environment (Colas et al., 2020; Zhou et al., 2023). Large-scale internet-scraped text and image data is a key enabler of current LLMs and text-to-image models (Schuhmann et al., 2022; Gadre et al., 2023; Penedo et al., 2023). However, data of comparable scale relating text with environments does not exist and human annotation is costly, leading to the question: *how can we learn LCAs given the scarcity of annotated data?*

Recently, foundation models (FMs) have emerged as promising tools to tackle this challenge. LLMs have been used to orchestrate predefined motion primitives (Huang et al., 2022; Zeng et al., 2022; Ichter et al., 2023), but the learning of such primitives also suffers from the lack of annotated data. Another approach is to use LLMs or vision-language models (VLMs) to define reward functions for a given text, either by source-code generation (Yu et al., 2023; Perez et al., 2023; Ma et al., 2023), or by image-text similarity evaluation (Mahmoudieh et al., 2022; Tam et al., 2022; Cui et al., 2022; Rocamonde et al., 2023; Adeniji et al., 2023). One may couple such rewards with single-task reinforcement learning (STRL), but such approaches are limited by the need to train a policy for each new task, making them too slow for real-time application. Training multi-task reinforcement learning (MTRL) agents is a natural alternative (Fan et al., 2022), which we explore in this paper. However, it is not obvious how to construct a training corpus of textual task descriptions that enables MTRL agents to generalize reliably to new tasks (Cobbe et al., 2020; Yu et al., 2020).

Therefore, this paper explores a novel decomposition of the problem of building LCAs into *VLM-based text-to-goal generation* and *goal-reaching*, as shown in Figure 1: we find a configuration such that rendered images of the environment in this configuration have high VLM scores for a given text; then we use a goal-conditioned reinforcement learning (GCRL) agent to reach that configuration[1]. In this paper, *configurations* capture enough of the state to render

[1]NAVER LABS Europe, Meylan [2]Institute of Intelligent Systems and Robotics, Sorbonne University, Paris. Correspondence to: Théo Cachet <theo.cachet@gmail.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1]Videos and an interactive demo can be found at https://europe.naverlabs.com/text2control

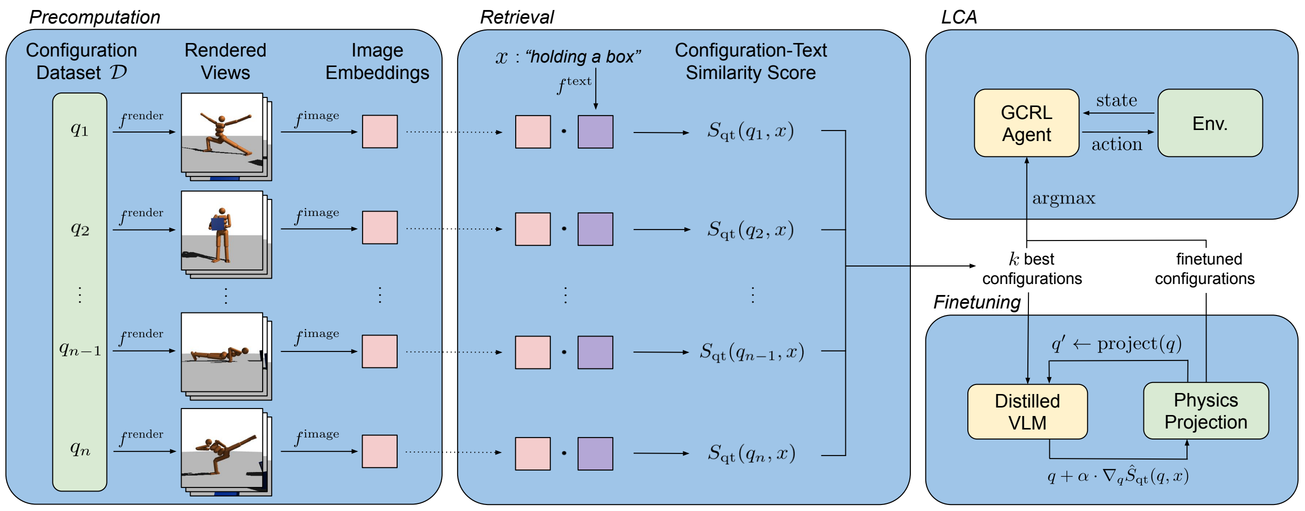


Figure 1. Overview of the proposed approach. We use rendering functions and a VLM image encoder to precompute embeddings of all configurations in a dataset. Given text x describing a new task, we embed it with the VLM text encoder, evaluate its cosine similarity with the precomputed image embeddings, select k highest-scoring configurations, and (optionally) finetune them using a distilled model. Finally, the best configuration is fed to a pretrained goal-conditioned agent to execute the task. Note that our approach can retrieve and finetune configurations for a text x without ever having seen that text before, and that the GCRL agent can reach a goal configuration without having been specifically trained for that goal; thus, our approach results in a *zero-shot* LCA.

images of the environment (e.g., current object positions of objects, but not velocities or past positions). The decomposition into text-to-goal and goal-reaching was previously proposed by Colas et al. (2020), but that work relies upon highly restricted algorithmic annotations to learn the text-to-goal component, rather than a VLM. Our decomposition has several advantages over the MTRL approach: it circumvents the problem of choosing a corpus of texts for MTRL; moreover, the reward function for GCRL is less oscillatory and faster to evaluate than the VLM score that might be used by MTRL. On the other hand, our decomposition precludes tasks with no natural final configuration, such as juggling or cycling, which we hope to study in future work.

We also explore several enhancements to the quality and speed of VLM-based LCAs. Whereas previous work has used the VLM score of a single view of an environment to determine the reward for a given state, we explore the use of *multiple viewpoints* to mitigate problems of occlusion and distance ambiguity inherent in a single 2D view. Moreover, we investigate the use of large datasets of diverse configurations with precomputed VLM embeddings, for rapid *retrieval of configurations* corresponding to a given text. These datasets may be also used to train *distilled models* (of the composition of rendering with the VLM image encoder) for rapid evaluation of VLM scores, accelerating both text-to-goal generation and the training of MTRL agents. We show that the derivatives of such distilled models with respect to configuration are better behaved than those of the original VLM score, and are well-suited to the finetuning of retrieved configurations. We evaluate the proposed methods on the Humanoid environment (Brockman et al., 2016), which we augment with feet and a cube, on a set of 256 textually defined tasks, using five state-of-the-art VLMs. To summarize the results:

- Our LCA, based on the novel decomposition into VLM-based text-to-goal generation and goal-reaching, attains higher returns than otherwise equivalent MTRL baselines, when performing zero-shot command execution, for 205 out of the 256 tasks.
- We quantify the viewpoint sensitivities of VLM scores, and give concrete examples showing that multiview evaluation avoids configurations that appear acceptable from one view but are actually pathological.
- Our distilled model reduces the computation time of VLM-based rewards by up to $40\,000\times$, while remaining sufficiently accurate that finetuning configurations using the distilled model increases the true VLM score.

The paper is structured as follows: we discuss related work (Section 2), present our methods (Section 3), experimentally evaluate them (Section 4), and suggest future work (Section 5). The Appendix presents full details about the methods, models, environment, tasks, rendering, as well as task-by-task results for the presented text-to-goal methods and LCAs.

2. Related Work

Central to our work is the challenge of *grounding language*: the process of linking language with an agent's observations and actions (Harnad, 1990). In this section, we first discuss the use of environment-specific annotations, LLMs and VLMs for grounding language and building LCAs. Then, we focus on text-to-goal generation methods.

Annotation. A natural approach to grounding is to gather textual annotations for an environment. For instance, state descriptions have been used to learn language-conditioned goal generators (Colas et al., 2020) and language-conditioned reward functions (Bahdanau et al., 2018; Nair et al., 2022). Additionally, state-sequence descriptions can be coupled with imitation learning (Stepputtis et al., 2020; Lynch & Sermanet, 2021; Chen et al., 2023) or with inverse reinforcement learning (Fu et al., 2019; Zhou & Small, 2021) to create LCAs. To reduce the cost of human annotation, some works generate annotations algorithmically (Jiang et al., 2019; Hill et al., 2020; Stooke et al., 2021), but so far such works are limited to simple tasks such as *"place X near Y"*.

Foundation models (FMs). Another way to circumvent costly human annotation is to exploit FMs. Given a textual task, LLMs have been used to write the source code of reward functions (Yu et al., 2023; Perez et al., 2023; Ma et al., 2023) or to orchestrate predefined skills (Huang et al., 2022; Ichter et al., 2023). LLMs have also been coupled with VLMs to leverage visual information (Zeng et al., 2022; Huang et al., 2023b; Ajay et al., 2023). However LLMs require environment-specific prompting and are prone to hallucination (Huang et al., 2023a).

Other works have also explored language grounding using VLMs. For instance, VLMs can be coupled with rendering functions to derive reward functions from natural language (Mahmoudieh et al., 2022; Fan et al., 2022; Rocamonde et al., 2023; Baumli et al., 2023), to pretrain language-conditioned policies (Adeniji et al., 2023), to derive extrinsic reward functions for exploration (Tam et al., 2022), and to detect task-completion (Du et al., 2023). Unfortunately, VLM-based reward functions are costly to evaluate, and they are highly oscillatory ('noisy'), as illustrated in Figure 3 of Adeniji et al. (2023), leading to slow and unreliable RL. Our approach also relies on VLMs, but we circumvent these difficulties by only using VLMs to find configurations with high VLM scores, as we now discuss.

Text-to-goal methods. Text-to-goal methods identify (sets of) states that align with a given textual description. These states may then be fed to goal-conditioned policies (Colas et al., 2020; Akakzia et al., 2021) or used to construct hybrid controllers (Raman et al., 2013), and thus to create LCAs.

With the emergence of large-scale text-to-text (Raffel et al., 2020), text-to-image (Rombach et al., 2022; Ramesh et al., 2022), text-to-audio (Copet et al., 2023) and text-to-video (Singer et al., 2022) models, it is interesting to investigate how FMs might serve as components of a text-to-goal procedure. Recently, Gao et al. (2023) proposed to leverage language-conditioned diffusion models to generate images representing goals. However, generating images that closely correspond to a given environment and instruction is challenging, even for state-of-the-art image editing techniques (Brooks et al., 2023; Dunlap et al., 2023). Moreover, additional steps are required to derive rewards or goal states from the resulting images: for instance, Gao et al. extend VICE (Fu et al., 2018) to infer rewards, which is computationally costly and error-prone. In contrast, our approach directly generates goal configurations, eliminating the need for image editing, and for such additional steps.

3. Methods

We address the overall problem of finding a language-conditioned policy: given text describing a (potentially previously unseen) task to be performed in an environment, the policy should result in configurations of that environment that correspond well to the given text. Our approach is to decompose this into two subproblems: finding configurations of the environment with high VLM scores for a given text; and designing goal-conditioned policies to reach such configurations. This decomposition has two main advantages over end-to-end training of LCAs. First, it is easy to generate a huge number of configurations to train a goal-conditioned policy, helping with generalization to new tasks, and avoiding the need to collect a large training set of task descriptions. Second, as grounding is independent of low-level control in our approach, it is possible to determine whether failures of the LCA are due to poor alignment of the goal with the task description, or to failure of the low-level controller to reach the goal.

The section begins with definitions relating configurations with VLM scores, before presenting methods for optimizing such scores that enable a range of speed-quality tradeoffs. Then we discuss the combination of text-to-goal methods with goal-conditioned policies, to craft zero-shot LCAs.

3.1. Definitions

Our approach selects *configurations* of an *environment* using *rendering functions* and *VLMs*, as we now explain.

Environment. The environment is modelled as a controlled Markov process

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \rho, \tilde{P}),$$

with state space $\mathcal{S}$, action space $\mathcal{A}$, initial state distribution ρ and transition distribution $\tilde{P}$. The controlled Markov process may be augmented with a reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ to define a Markov decision process (MDP). In this work, we define various reward functions, using distances to goal configurations, or VLM scores for a given text (in our baselines).

Configurations. Each state of $\mathcal{S}$ is associated with a specific *configuration* $q \in \mathcal{Q}$, which captures the state dimensions relevant to rendering images of the environment. For instance, a configuration might consist of the angles or positions of an environment's bodies, but not their velocities. As we use gradient methods to optimize configurations, we assume the configuration space $\mathcal{Q}$ can be represented as a subset of a real Euclidean space. Typically, not all configurations are admissible: there may be inequality constraints corresponding to the requirement that objects do not interpenetrate (unilateral constraints) or to safety requirements. We denote the admissible subset of configurations by $\mathcal{Q}^{\text{a}}$.

Rendering functions. A *rendering function* $f^{\text{render}} : \mathcal{Q} \to \mathcal{I}$ maps configurations to images of the environment. Suitable rendering functions can be found in MuJoCo (Todorov et al., 2012) and OpenGL (Neider et al., 1993).

VLMs. A VLM, such as CLIP (Radford et al., 2021), typically consists of

1. An image encoder $f^{\text{image}} : \mathcal{I} \to \mathbb{R}^d$ that maps images to an embedding space of dimension d; and
2. A text encoder $f^{\text{text}} : \mathcal{T} \to \mathbb{R}^d$ that maps texts to the same embedding space. The space of texts $\mathcal{T}$ consists of finite strings on a finite vocabulary.

We assume that the outputs of these encoders are normalized so that $\|f^{\text{image}}(\cdot)\| = 1$ on $\mathcal{I}$ and $\|f^{\text{text}}(\cdot)\| = 1$ on $\mathcal{T}$. The *image-text similarity score* (or *VLM score*) for a given image $I \in \mathcal{I}$ and text $x \in \mathcal{T}$ is then defined as the cosine similarity of their embeddings:

$$S_{\text{it}}(I, x) := f^{\text{image}}(I) \cdot f^{\text{text}}(x). \tag{1}$$

Configuration-text score. Composing a rendering function with the image-text similarity score enables us to define a *configuration-text similarity score* for a given configuration $q \in \mathcal{Q}$ and text $x \in \mathcal{T}$:

$$S^{\text{sv}}_{\text{qt}}(q, x) := S_{\text{it}}(f^{\text{render}}(q), x), \tag{2}$$

where 'sv' emphasizes that this is for a single view. To resolve ambiguities inherent in a single 2D view, we propose to extend this definition to a *multiview configuration-text similarity score* by averaging the similarity scores for multiple rendering functions $f^{\text{render}}_1, \ldots, f^{\text{render}}_m$:

$$\begin{aligned} S_{\text{qt}}(q, x) &:= \frac{1}{m} \sum_{k=1}^{m} S_{\text{it}}(f^{\text{render}}_k(q), x) \\ &= f^{\text{config}}(q) \cdot f^{\text{text}}(x), \end{aligned} \tag{3}$$

where the *configuration embedding* $f^{\text{config}} : \mathcal{Q} \to \mathbb{R}^d$ is defined by

$$f^{\text{config}}(q) := \frac{1}{m} \sum_{k=1}^{m} f^{\text{image}} \circ f^{\text{render}}_k(q). \tag{4}$$

Distilled model. Given multiple rendering functions and billion-parameter VLMs, evaluating the configuration-text similarity score can be costly. Therefore, we propose to *distill* the configuration embedding into a neural network

$$\hat{f}^{\text{config}}(q) \approx f^{\text{config}}(q). \tag{5}$$

Not only is the distilled model $\hat{f}^{\text{config}}$ faster than f^{config}, it is also readily differentiated with respect to the configuration: this proves useful when optimizing scores with respect to the configuration. In contrast, the gradients of f^{render}_k and hence f^{config} may not be defined; and even if they are, the embedding f^{config} may be highly oscillatory (see Figure 23 in the Appendix), making its gradients of questionable utility.

We use the distilled model in three ways: to sample a diverse dataset for retrieval of configurations; to finetune the resulting configurations; and to train STRL and MTRL baselines.

3.2. Text-to-Goal Generation Methods

Given a text x, we wish to find an admissible configuration q^* that maximizes the configuration-text similarity score:

$$q^* \in \underset{q \in \mathcal{Q}^{\text{a}}}{\operatorname{argmax}}\ f^{\text{config}}(q) \cdot f^{\text{text}}(x).$$

As the similarity score can be highly multimodal and costly to evaluate, we adopt a three-step approach to optimizing it for a given text, which involves: (i) retrieving high-scoring configurations from a dataset of precomputed configuration embeddings; (ii) starting from those configurations, performing gradient ascent on an approximate version of the similarity score, based on the distilled model of the configuration embedding; and (iii) selecting from the resulting configurations using the exact similarity score, based on the VLM. These steps are explained in detail below. Optionally, one might stop immediately after step (i), returning the single best configuration in the dataset; or after step (ii), returning the best configuration according to the approximate score.

3.2.1. Retrieving Configurations

As the configuration embedding f^{config} is independent of the text, one way to mitigate the cost of optimizing the configuration-text similarity is to work with a dataset of configurations $\mathcal{D} := \{q_1, q_2, \ldots, q_n\} \subset \mathcal{Q}^{\text{a}}$ with *precomputed* configuration embeddings $f^{\text{config}}(q)$ for $q \in \mathcal{D}$. Given a text x, one may then retrieve a configuration with a high configuration-text similarity score, simply by taking dot products with those precomputed embeddings:

$$q^{\text{retrieved}} \in \operatorname*{argmax}_{q \in \mathcal{D}} f^{\text{config}}(q) \cdot f^{\text{text}}(x). \tag{6}$$

The effectiveness of this retrieval method hinges on the choice of the retrieval dataset $\mathcal{D}$. To ensure high-scoring configurations for *any* text, dataset $\mathcal{D}$ must encompass diverse configurations. We therefore propose the following three dataset-generation methods aiming for diversity, while ensuring admissibility of the resulting configurations (full details are given in Appendix B):

Random policy dataset. This dataset consists of configurations resulting from a random policy interacting with the environment. To encourage diversity, the random policy samples actions uniformly over the action space.

Uniform sampling dataset. This dataset is generated by uniformly sampling from the configuration space $\mathcal{Q}$ and projecting them onto admissible configurations $\mathcal{Q}^{\text{a}}$.

Embedding-diversity dataset. This dataset is created by focusing on the diversity of configuration embeddings, using the distilled model (5). We optimize a set of configurations by minimizing the maximum cosine similarity between embeddings of distinct configurations with the following loss:

$$L(q_1, \ldots, q_n) = \frac{1}{n} \sum_{i=1}^{n} \max_{j \in [n] \setminus \{i\}} \hat{f}^{\text{config}}(q_j) \cdot \hat{f}^{\text{config}}(q_i).$$

3.2.2. Finetuning Configurations

Even a huge retrieval dataset may lack configurations aligned with a given text. Therefore, we propose a method to finetune retrieved configurations. As the exact configuration-text similarity score is costly and may not be differentiable, we attempt to maximize a surrogate $\hat{S}_{\text{qt}}$ for that score, based on the distilled model (5):

$$\hat{S}_{\text{qt}}(q, x) := \hat{f}^{\text{config}}(q) \cdot f^{\text{text}}(x). \tag{7}$$

By the product rule, the gradient of this surrogate score is

$$\nabla_q \hat{S}_{\text{qt}}(q, x) = \nabla_q \hat{f}^{\text{config}}(q) \cdot f^{\text{text}}(x).$$

As some configurations may not be admissible, we assume a projection operator $\mathcal{P}_{\mathcal{Q}^{\text{a}}} : \mathcal{Q} \to \mathcal{Q}^{\text{a}}$ is available, which maps configurations to nearby admissible configurations. In our experiments, we implement $\mathcal{P}_{\mathcal{Q}^{\text{a}}}$ with one step of the MuJoCo physics engine. (This step does no dynamics, it just recovers from any interpenetrations.) We use this projection to perform projected gradient ascent, with the update

$$q^{(j+1)} = \mathcal{P}_{\mathcal{Q}^{\text{a}}}\left(q^{(j)} + \alpha \cdot \nabla_q \hat{S}_{\text{qt}}(q^{(j)}, x)\right), \tag{8}$$

where α is the learning rate. The full procedure is presented in Appendix E. In practice, both the gradient and projection calculations are highly parallelizable, allowing concurrent optimization of multiple solutions.

3.3. Zero-Shot LCA

To craft an LCA, we feed a text x to one of the text-to-goal methods described above, resulting in a goal configuration q_x^{goal}. Then we feed that goal to a task-agnostic goal-conditioned policy π, which takes actions distributed as $a \sim \pi(\cdot|s, q_x^{\text{goal}})$ when in state s. We train this policy on a collection of goals with GCRL (Colas et al., 2022; Liu et al., 2022), using a reward based on the Euclidean distance between the current configuration and a goal configuration q:

$$R(s, a|q) = \|\varphi_{\mathcal{Q}}(s) - q\| - \|\varphi_{\mathcal{Q}}(P(s, a)) - q\|, \tag{9}$$

where $\varphi_{\mathcal{Q}} : \mathcal{S} \to \mathcal{Q}$ is the mapping from state to configuration, with configurations seen as Euclidean vectors, and P gives the next state (assuming transitions distributed as $\tilde{P}$ are deterministic). At test time, we simply condition the GCRL agent on a goal configuration to execute the task, without additional training or pre-processing.

4. Experiments

In this section, we evaluate our text-to-goal methods and LCAs to address the following questions: **(Q1)** How does the choice of dataset-sampling method affect the performance of VLM-based configuration retrieval? **(Q2)** What is the impact of using multiple viewpoints to assess configurations in 3D environments? **(Q3)** What is the speed-quality trade-off of the proposed text-to-goal methods? **(Q4)** How do the proposed LCAs compare with STRL and MTRL baselines?

4.1. Evaluation Benchmark

Environment and rendering functions. We evaluate our approach on the Humanoid environment from OpenAI's Gym framework (Brockman et al., 2016), which we augment with feet and with a cube, enabling stable standing and agent-object interaction. We create three MuJoCo rendering functions *front view*, *right view* and *left view* (Figure 2) to

Evaluator Model	Random Policy Dataset	Uniform Sampling Dataset	Embedding Diversity Dataset
EVA02-E-14+ (LAION2B)	0.6132	0.6128	**0.6260**
EVA02-E-14 (LAION2B)	0.3416	0.3403	**0.3496**
ViT-H-14 (DFN5B)	0.4016	0.4011	**0.4102**
ViT-H-14 (MetaCLIP)	0.4114	0.4092	**0.4197**
ViT-bigG-14 (DataComp1B)	0.2700	0.2681	**0.2771**

Table 1. Average single-view configuration-text scores of retrieved configurations by dataset, evaluated with different VLMs. EVA02-E-14(+), DFN5B, MetaCLIP and DataComp1B are from Sun et al. (2023), Xu et al. (2023) and Gadre et al. (2023) respectively.

render configurations of that environment. Full details are in Appendix A.

Tasks. We evaluate our work on 256 diverse tasks, which are descriptions of environment configurations in natural language. The list of tasks and the process used to generate it are described in Appendix I. The tasks range from simple concepts (*"standing up"*), to compositional relationships (*"sitting on top of a box"*) and abstract concepts (*"standing up in Usain Bolt's celebration pose"*).

4.2. Dataset-Sampling Evaluation (Q1)

We sample three datasets of 2.5×10^6 configurations using the three dataset-sampling methods described in Section 3.2.1 and precompute their configuration embeddings (4) with the front-view rendering function and the EVA-02-E-14+ VLM (Sun et al., 2023). Then for all tasks, we retrieve the configuration (6) with the highest configuration-text score from each dataset.

Table 1 presents the configuration-text scores of the retrieved configurations as assessed by various VLMs, averaged over all tasks. We observe that the configurations retrieved from the embedding-diversity dataset attain the highest scores for all VLMs. Therefore we use that dataset-sampling method in the remaining experiments. Further results comparing different dataset sizes and using multiview scores are discussed in Appendix B.

4.3. Multiview Configuration-Text Score (Q2)

We now study the impact of using multiple views to retrieve configurations, as in equation (3).

Table 2 compares configurations retrieved from the embedding-diversity dataset using three views (at $0°, \pm 45°$) with configurations retrieved using the front view ($0°$) only. These two sets of configurations are evaluated with single-view configuration-text scores for different viewpoints. We include a *mid-left* view at $-22.5°$ to assess generalization

Evaluator Model	Retrieval # Views	Front View	Mid-Left View	Left View
EVA02-E-14+ (LAION2B)	1	**0.6260**	0.6050	0.5835
	3	0.6147	**0.6138**	**0.6104**
EVA02-E-14 (LAION2B)	1	**0.3496**	0.3340	0.3164
	3	0.3442	**0.3435**	**0.3386**
ViT-H-14 (DFN5B)	1	**0.4102**	0.3992	0.3816
	3	0.4092	**0.4053**	**0.3999**
ViT-H-14 (MetaCLIP)	1	**0.4197**	0.4131	0.4021
	3	0.4194	**0.4202**	**0.4172**
ViT-bigG-14 (DataComp1B)	1	**0.2771**	0.2661	0.2554
	3	0.2769	**0.2734**	**0.2722**

Table 2. Single-view configuration-text scores, evaluated by various VLMs, of configurations retrieved using one view and three views from the embedding-diversity dataset, seen from different viewpoints. Appendix C gives results for other viewpoints.

to rendering functions distinct from those used for retrieval. When evaluated on the *front* view, the configurations retrieved using the front view only attain higher scores than those retrieved using three views. But when evaluated from *other* views, their scores decrease significantly. In contrast, the configurations retrieved using three views exhibit little variation in score across the viewpoints tested.

Figure 2 provides some insight into the lack of robustness of single-view retrieval, illustrating six tasks for which a high single-view configuration-text score is misleading. We identify the following three reasons for these failures:

- *Occlusion.* At times, some components hide other components. For instance, in the top three rows of column 1 of Figure 2, the right leg of the humanoid is behind its body. This is not apparent in the front view (top row), but it becomes apparent in the left and right views. Using three views to evaluate configurations mitigates this issue.
- *Distance ambiguity.* Based on a single view, the distance between objects in a scene is ambiguous, even for a human. In columns 3 and 4, the humanoid and the cube appear close to each other in the front view (top row), but the side views (second and third rows) reveal that they are actually far apart.
- *Stability.* While the top row of columns 5 and 6 appears to show configurations that correspond well to the specified tasks, the left and right views reveal that the positions are unstable. While our text-to-goal method does not explicitly aim for stability, stability emerges from optimizing the multiview configuration-text score.

We conclude that evaluating configurations using multiple views mitigates some of the problems arising from the lack

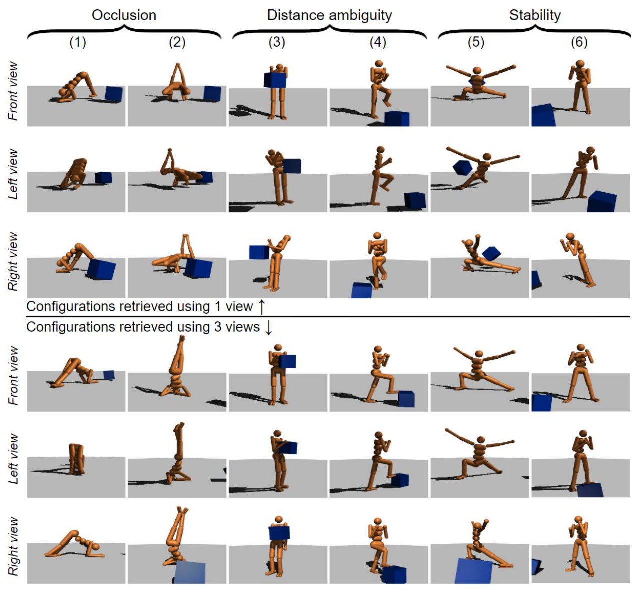


Figure 2. Each column represents a task: (1) *"downward-facing dog yoga pose"*, (2) *"headstand"*, (3) *"holding a blue box"*, (4) *"box step-up"*, (5) *"warrior yoga pose"* and (6) *"boxing guard"*. Within each column, the top three rows show the front, left and right views of the best configuration from the embedding-diversity dataset, retrieved using the front view only. The bottom three rows show the best configuration retrieved using the three views.

of information of single 2D images. While using multiple views linearly increases the computational cost of precomputing the configuration embeddings, that improvement comes at no extra cost at retrieval time.

4.4. Configuration Finetuning (Q3, Quality)

We now study the optimization of retrieved configurations. First, we use the configurations from the embedding-diversity dataset and their precomputed embeddings to train a distilled model (as detailed in Appendix D). Then for each task, we use that model to finetune the 256 top-scoring configurations (based on the multiview score) from the embedding-diversity dataset (as detailed in Appendix E). We compare two options after finetuning: either we return the configuration with the highest approximate score (7) based on the distilled model; or we evaluate all finetuned configurations with EVA02-E-14+ and select the one with the highest exact multiview score (3). The choice of hyperparameters for finetuning is discussed in Appendix E: one reason for using 256 configurations is that it corresponds to one batch of VLM evaluations.

Table 3 presents results of finetuning, indicating that the proposed finetuning methods further improve the configuration-text alignment of retrieved configurations, even when evaluated by other VLMs. Figure 3 illustrates retrieved configurations and their finetuned counterparts, showing that even

Evaluator Model	Retrieval Only	Retrieval+ Finetuning	Retrieval+ Finetuning+ Selection
EVA02-E-14+ (LAION2B)	0.6265	0.6289	**0.6343**
EVA02-E-14 (LAION2B)	0.3506	0.3528	**0.3564**
ViT-H-14 (DFN5B)	0.4187	0.4224	**0.4229**
ViT-H-14 (MetaCLIP)	0.4226	0.4243	**0.4255**
ViT-bigG-14 (DataComp1B)	0.2771	0.2795	**0.2800**

Table 3. Multiview configuration-text scores of retrieved configurations, finetuned configurations, and finetuned configurations selected by EVA02-E-14+. These scores are evaluated by various VLMs and averaged over the 256 tasks.

small numerical increases in the configuration-text score can correspond to significant changes in configuration. Finetuned configurations are sometimes more convincing (and hardly ever less convincing) to a human viewer.

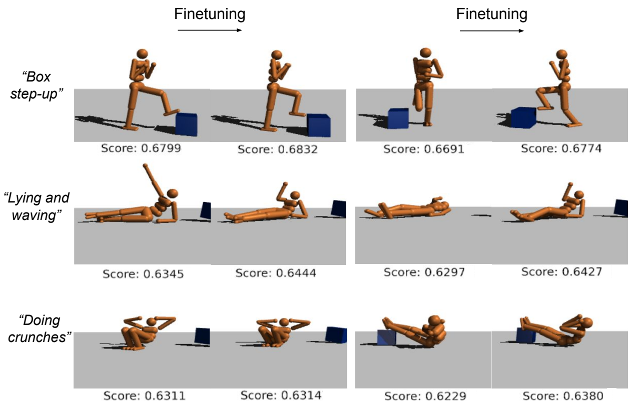


Figure 3. Each row shows a single task. Columns 1 and 3 show the top-1 and another of the top-20 retrieved configurations; columns 2 and 4 show their finetuned counterparts (without VLM selection).

Finally, Appendix E studies the effect of the number of finetuning steps.

4.5. Time Efficiency (Q3, Speed)

Evaluating 256 three-view configuration embeddings with our distilled model takes 4.8×10^{-4} s, whereas computing those embeddings with EVA02-E-14+ takes 21.1 s (for the computing infrastructure described in Appendix F). Thus, the distilled model reduces the computation time of configuration embeddings by $40\,000\times$.

The timings of our text-to-goal methods are as follows: embedding the text takes 0.013 s, retrieving high-scoring configurations with precomputed embeddings (by brute force, from a dataset of 2.5×10^6 samples) takes 0.015 s; and the optional stages of finetuning and selecting the highest-scoring configuration (by applying EVA02-E-14+ to three views, corresponding to three batches) take 2.8 s and 21.1 s respectively.[2]

This compares favourably with approaches based on goal-image generation (Gao et al., 2023), which would require $\geq 3 \times 20$ s were they to generate three goal images to overcome the ambiguities of 2D images. It also compares favourably with using LLMs to generate source code specifying goals. For instance, LLAMA-2-70B (Touvron et al., 2023) generates about 10 tokens per second on our infrastructure, and requires about 15 s to generate a Humanoid configuration. In conclusion, our text-to-goal approaches are fast enough for real-time language-guided control of humanoid agents.

4.6. Language-Conditioned Agents (Q4)

We now detail our baselines and the GCRL policy. Then we compare the resulting LCAs.

Observation and action spaces. Each of the LCAs receives a 343-dimensional observation vector. This includes the current position, orientation and velocity of each component of the Humanoid and the cube, along with the timestep (as detailed in Appendix A). For MTRL, this observation is augmented with the text embedding of the task. For GCRL, it is augmented with the goal configuration. The agents take 19-dimensional actions, corresponding to actuator torques.

MTRL and STRL baselines. Previous works (Mahmoudieh et al., 2022; Rocamonde et al., 2023; Fan et al., 2022; Adeniji et al., 2023) use the single-view configuration-text score to define the reward function

$$R_x^{\text{past work}}(s,a) = S_{\text{qt}}(\varphi_Q(s), x) \tag{10}$$

for action a, in state s, with text x. We propose three improvements. First, we use three-view configuration-text scores, which improve robustness as shown in Section 4.3. Second, we define the *VLM reward* as the time difference of configuration-text scores:

$$R_x(s,a) = S_{\text{qt}}(\varphi_Q(P(s,a)), x) - S_{\text{qt}}(\varphi_Q(s), x). \tag{11}$$

This reward encourages policies to attain high final scores, rather than to maintain high average scores as discussed in Appendix G. Finally, we define the *approximate VLM reward*, using the distilled model (the same one that we used for finetuning):

$$\hat{R}_x(s,a) = \hat{S}_{\text{qt}}(\varphi_Q(P(s,a)), x) - \hat{S}_{\text{qt}}(\varphi_Q(s), x). \tag{12}$$

This reward is up to $40\,000\times$ faster to evaluate and less oscillatory than R_x.

[2]Our current implementation is based on a CPU version of MuJoCo: finetuning may be much faster with a GPU version. Selecting the highest-scoring configurations could also be much faster: for instance, one evaluation of EVA02-E-14+ requires 2362B FLOPs, whereas SigLIP (Zhai et al., 2023) requires 233B.

Unlike previous work, which only trains STRL agents, we also train MTRL agents. Both STRL and MTRL agents use reward $\hat{R}_x$. The MTRL agents learn a multi-task policy $\pi(a|s, f^{\text{text}}(x))$, which is conditioned on the VLM text-embedding. This enables zero-shot execution of new tasks, so it is directly comparable with the proposed LCA based on text-to-goal generation. To compare the performance of MTRL on test and training tasks, we randomly partition the 256 tasks into two sets of 128, and train one MTRL agent on each set.

GCRL agent. We train the GCRL agent to maximize reward (9) using goals sampled uniformly from the embedding-diversity dataset. At test time, we condition the agent on configurations retrieved and potentially fine-tuned using the methods of Section 3.2.

Architectures and training. The MTRL, STRL and GCRL agents have identical architectures (as far as possible given their different inputs). All are trained with proximal policy optimization (Schulman et al., 2017) with similar hyperparameters. Full details are given in Appendix F.

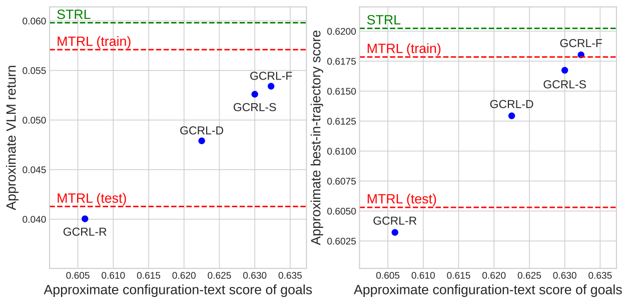


Figure 4. Approximate VLM returns and approximate best-in-trajectory configuration-text scores for the STRL and MTRL baselines, and for the GCRL agent given four types of goal configuration: retrieval from the random policy dataset (*GCRL-R*); and retrieval from the diversity dataset (*GCRL-D*), plus finetuning (*GCRL-F*), plus selection based on the exact score (*GCRL-S*). *MTRL (train)* and *MTRL (test)* are for the baseline models evaluated on their training and test tasks respectively. The metrics are averaged over all 256 tasks, with 10 episodes per task.

Comparison. Figure 4 compares the approximate VLM returns, given by discounted sums of reward (12), and the approximate best-in-trajectory configuration-text scores

$$\max\{\hat{S}_{\text{qt}}(\varphi_Q(s), x) : s \in \text{trajectory}\}$$

for the STRL and MTRL baselines, and for the GCRL agent given four different types of goal. Here we choose to show the approximate metrics because the baselines were trained to optimize the approximate VLM return. Even though the GCRL agents were not trained to optimize the VLM return, GCRL-D, GCRL-F and GCRL-S (acronyms defined in the caption) all attain higher approximate VLM returns than MTRL (test).

This demonstrates that LCAs based on a decomposition into goal-generation and goal-attainment can generalize more effectively to previously unseen tasks than MTRL agents. Moreover, the STRL agents achieve an average approximate best-in-trajectory score of 0.6202, whereas the finetuned goal configurations have an average score of 0.6324. Thus, if the GCRL agent were able to accurately reach its goals, then it would outperform the STRL agents on that metric.

Appendix G presents analogous results using *exact* configuration-text scores, which do not differ greatly differ from those for approximate scores. It also presents results for each task individually. In summary, the exact and approximate VLM returns of GCRL-F exceed those of MTRL (test) in 198 and 203 of the 256 tasks respectively; while the exact VLM return of GCRL-S exceeds that of MTRL (test) in 205 tasks.

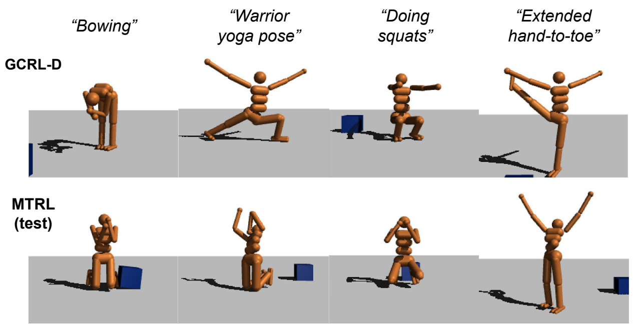


Figure 5. Front-view of the best-in-trajectory configuration (with respect to the approximate VLM score) reached by GCRL-D and by the MTRL (test) baseline in a single rollout. Analogous figures for all LCAs on all 256 tasks are in Appendix J.2.

Generalization. As illustrated in Figure 5, the generalization of GCRL-D to new tasks is often superior to that of MTRL (test). We propose the following two hypotheses explaining this observation:

1. The training set of GCRL is more helpful than that of MTRL: it is larger and it is optimized for diversity. It is larger as training tasks for GCRL only consist of configurations to reach, and these are simpler to collect than the texts required for MTRL training tasks. Optimizing for diversity is clearly beneficial: indeed, Figure 4 shows that GCRL-R (random policy) does worse than MTRL-test, whereas GCRL-D (diversity) does better.

2. GCRL training tasks are easier to learn than those of MTRL. Indeed, the GCRL agent only needs to learn how to reach already-found high-scoring configurations, whereas the MTRL agent must both find high-scoring configurations and learn to reach them. Moreover, Appendix H.2 demonstrates that the VLM score is highly oscillatory, which could perturb RL training. While Appendix D shows that the approximate VLM score used for MTRL is smoother than the true VLM score, it is still much more oscillatory than the Euclidean distance used for GCRL.

These advantages come with a trade-off. While MTRL directly optimizes the approximate VLM return, GCRL uses configurations as a proxy to maximize that return. However, some configurations may be hard-to-reach, and proximity to a goal configuration in terms of Euclidean distance does not necessarily translate to a high approximate VLM score, as discussed in Appendix G.5.

5. Limitations and Further Work

We would like to extend the range of tasks that may be performed to include *compositional relationships* between a diverse set of objects. As the current generation of VLMs is limited in its ability to assess such relationships, as discussed in Appendix H.2, we await a new generation of VLMs to enable such work. Also, our current approach is limited to static configurations: in future, it will be interesting to extend the approach to *dynamic tasks*.

Acknowledgements

We thank Quan Sun *et al.* as well as the OpenCLIP community for releasing pretrained state-of-the-art VLMs, enabling such research. We also thank Jean-Michel Renders, Tomi Silander, Diane Larlus and David Emukpere and the reviewers for their many helpful comments.

Impact Statement

Positive impacts. The development of language-conditioned agents (LCAs) will further fluidify human-technology interaction. Given the convenience of language as a medium for humans to express desires and commands, this will likely result in an increase in such interactions. The possibility of steering technologies easily will empower every potential user, including those without technical expertise. LCAs are particularly promising as they might enhance the independence of individuals with mobility limitations, such as elderly or paraplegic persons. They may also be valuable for fine-grained control of robots in hazardous environments, such as radioactive or extraterrestrial terrains.

Negative impacts. The proposed technology can be used to enable language-conditioned design and interaction with virtual environments, such as games, potentially making those environments more addictive, possibly causing users to become dependent upon them, lose touch with reality and become isolated. Additionally, a person of malicious intent could specify tasks that might be harmful (e.g., kick something hard) or insulting (e.g., make some kind of insulting gesture) to the LCA, and use it to generate corresponding images or to control a physical robot. As with any work that might be embodied in a physical robot, there is a risk of physical harm to objects and entities in that robot's immediate vicinity: the training of the RL agents would need to be adapted to take safety factors into account; the agents would need to be trained/finetuned on real-world data; and operators would need to be trained on appropriate safety guidelines.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. GPT-4 technical report. *arXiv:2303.08774*, 2023.

Adeniji, A., Xie, A., Sferrazza, C., Seo, Y., James, S., and Abbeel, P. Language reward modulation for pretraining reinforcement learning. *arXiv:2308.12270*, 2023.

Ajay, A., Han, S., Du, Y., Li, S., Gupta, A., Jaakkola, T., Tenenbaum, J., Kaelbling, L., Srivastava, A., and Agrawal, P. Compositional foundation models for hierarchical planning. *arXiv:2309.08587*, 2023.

Akakzia, A., Colas, C., Oudeyer, P., Chetouani, M., and Sigaud, O. Grounding language to autonomously-acquired skills via goal generation. In *International Conference on Learning Representations*, 2021.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. *Advances in Neural Information Processing Systems*, 30, 2017.

Bahdanau, D., Hill, F., Leike, J., Hughes, E., Hosseini, A., Kohli, P., and Grefenstette, E. Learning to understand goal specifications by modelling reward. *arXiv:1806.01946*, 2018.

Baumli, K., Baveja, S., Behbahani, F., Chan, H., Comanici, G., Flennerhag, S., Gazeau, M., Holsheimer, K., Horgan, D., Laskin, M., et al. Vision-language models as a source of rewards. *arXiv:2312.09187*, 2023.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI Gym. *arXiv:1606.01540*, 2016.

Brooks, T., Holynski, A., and Efros, A. A. InstructPix2Pix: Learning to follow image editing instructions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 18392–18402, 2023.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv:2303.12712*, 2023.

Chen, S., Garcia, R., Schmid, C., and Laptev, I. PolarNet: 3D point clouds for language-guided robotic manipulation. *arXiv:2309.15596*, 2023.

Chu, E., Lin, S.-Y., and Chen, J.-C. Video ControlNet: Towards temporally consistent synthetic-to-real video translation using conditional image diffusion models. *arXiv:2305.19193*, 2023.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging procedural generation to benchmark reinforcement learning. In *International Conference on Machine Learning*, pp. 2048–2056, 2020.

Colas, C., Akakzia, A., Oudeyer, P.-Y., Chetouani, M., and Sigaud, O. Language-conditioned goal generation: a new approach to language grounding for RL. *arXiv:2006.07043*, 2020.

Colas, C., Karch, T., Sigaud, O., and Oudeyer, P.-Y. Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. *Journal of Artificial Intelligence Research*, 74:1159–1199, 2022.

Copet, J., Kreuk, F., Gat, I., Remez, T., Kant, D., Synnaeve, G., Adi, Y., and Défossez, A. Simple and controllable music generation. *arXiv:2306.05284*, 2023.

Cui, Y., Niekum, S., Gupta, A., Kumar, V., and Rajeswaran, A. Can foundation models perform zero-shot task specification for robot manipulation? In *Learning for Dynamics and Control Conference*, pp. 893–905, 2022.

Du, Y., Konyushkova, K., Denil, M., Raju, A., Landon, J., Hill, F., de Freitas, N., and Cabi, S. Vision-language models as success detectors. *arXiv:2303.07280*, 2023.

Dunlap, L., Umino, A., Zhang, H., Yang, J., Gonzalez, J. E., and Darrell, T. Diversify your vision datasets with automatic diffusion-based augmentation. *CoRR*, abs/2305.16289, 2023.

Fan, L., Wang, G., Jiang, Y., Mandlekar, A., Yang, Y., Zhu, H., Tang, A., Huang, D.-A., Zhu, Y., and Anandkumar, A. MineDojo: Building open-ended embodied agents with internet-scale knowledge. *arXiv:2206.08853*, 2022.

Fu, J., Singh, A., Ghosh, D., Yang, L., and Levine, S. Variational inverse control with events: A general framework for data-driven reward definition. *Advances in Neural Information Processing Systems*, 31, 2018.

Fu, J., Korattikara, A., Levine, S., and Guadarrama, S. From language to goals: Inverse reinforcement learning for vision-based instruction following. *arXiv:1902.07742*, 2019.

Gadre, S. Y., Ilharco, G., Fang, A., Hayase, J., Smyrnis, G., Nguyen, T., Marten, R., Wortsman, M., Ghosh, D., Zhang, J., et al. DataComp: In search of the next generation of multimodal datasets. *arXiv:2304.14108*, 2023.

Gao, J., Hu, K., Xu, G., and Xu, H. Can pre-trained text-to-image models generate visual goals for reinforcement learning? *arXiv:2307.07837*, 2023.

Harnad, S. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346, 1990.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (GELUs). *arXiv:1606.08415*, 2016.

Hill, F., Mokra, S., Wong, N., and Harley, T. Human instruction-following with deep reinforcement learning via transfer-learning from text. *arXiv:2005.09382*, 2020.

Hsieh, C.-Y., Zhang, J., Ma, Z., Kembhavi, A., and Krishna, R. Sugarcrepe: Fixing hackable benchmarks for vision-language compositionality. *arXiv:2306.14610*, 2023.

Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv:2311.05232*, 2023a.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pp. 9118–9147, 2022.

Huang, W., Wang, C., Zhang, R., Li, Y., Wu, J., and Fei-Fei, L. Voxposer: Composable 3D value maps for robotic manipulation with language models. *arXiv:2307.05973*, 2023b.

Ichter, B., Brohan, A., Chebotar, Y., Finn, C., Hausman, K., Herzog, A., Ho, D., Ibarz, J., Irpan, A., Jang, E., Julian, R., Kalashnikov, D., Levine, S., Lu, Y., Parada, C., Rao, K., Sermanet, P., Toshev, A. T., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Yan, M., Brown, N., Ahn, M., Cortes, O., Sievers, N., Tan, C., Xu, S., Reyes, D., Rettinghouse, J., Quiambao, J., Pastor, P., Luu, L., Lee, K.-H., Kuang, Y., Jesmonth, S., Joshi, N. J., Jeffrey, K., Ruano, R. J., Hsu, J., Gopalakrishnan, K., David, B., Zeng, A., and Fu, C. K. Do as I can, not as I say: Grounding language in robotic affordances. In *Conference on Robot Learning*, pp. 287–318, 2023.

Jiang, Y., Gu, S. S., Murphy, K. P., and Finn, C. Language as an abstraction for hierarchical deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Kato, H., Beker, D., Morariu, M., Ando, T., Matsuoka, T., Kehl, W., and Gaidon, A. Differentiable rendering: A survey. *arXiv:2006.12057*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

Lewis, M., Nayak, N. V., Yu, P., Yu, Q., Merullo, J., Bach, S. H., and Pavlick, E. Does CLIP bind concepts? Probing compositionality in large image models. *arXiv:2212.10537*, 2022.

Liu, M., Zhu, M., and Zhang, W. Goal-conditioned reinforcement learning: Problems and solutions. *arXiv:2201.08299*, 2022.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv:1711.05101*, 2017.

Lynch, C. and Sermanet, P. Language conditioned imitation learning over unstructured data. *arXiv:2005.07648*, 2021.

Ma, Y. J., Liang, W., Wang, G., Huang, D.-A., Bastani, O., Jayaraman, D., Zhu, Y., Fan, L., and Anandkumar, A. Eureka: Human-level reward design via coding large language models. *arXiv:2310.12931*, 2023.

Mahmoudieh, P., Pathak, D., and Darrell, T. Zero-shot reward specification via grounded natural language. In *International Conference on Machine Learning*, pp. 14743–14752, 2022.

Nair, S., Mitchell, E., Chen, K., ichter, b., Savarese, S., and Finn, C. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*, 2022.

Neider, J., Davis, T., and Woo, M. *OpenGL Programming Guide*, volume 478. Addison-Wesley Reading, MA, 1993.

Pardo, F., Tavakoli, A., Levdik, V., and Kormushev, P. Time limits in reinforcement learning. In *International Conference on Machine Learning*, pp. 4045–4054. PMLR, 2018.

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., and Launay, J. The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. *arXiv:2306.01116*, 2023.

Perez, J., Proux, D., Roux, C., and Niemaz, M. LARG, language-based automatic reward and goal generation. *arXiv:2306.10985*, 2023.

Plappert, M., Sampedro, R., Xu, T., Akkaya, I., Kosaraju, V., Welinder, P., D'Sa, R., Petron, A., de Oliveira Pinto, H. P., Paino, A., Noh, H., Weng, L., Yuan, Q., Chu, C., and Zaremba, W. Asymmetric self-play for automatic goal discovery in robotic manipulation. *arXiv:2101.04882*, 2021.

Radenovic, F., Dubey, A., Kadian, A., Mihaylov, T., Vandenhende, S., Patel, Y., Wen, Y., Ramanathan, V., and Mahajan, D. Filtering, distillation, and hard negatives for vision-language pre-training. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2023.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763, 2021.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21 (1):5485–5551, 2020.

Raman, V., Lignos, C., Finucane, C., Lee, K. C. T., Marcus, M. P., and Kress-Gazit, H. Sorry Dave, I'm afraid I can't do that: Explaining unachievable robot tasks using natural language. In *Robotics: Science and Systems IX*, 2013.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with CLIP latents. *arXiv:2204.06125*, 1(2):3, 2022.

Rocamonde, J., Montesinos, V., Nava, E., Perez, E., and Lindner, D. Vision-language models are zero-shot reward models for reinforcement learning. *arXiv:2310.12921*, 2023.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al. LAION-5B: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35: 25278–25294, 2022.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017.

Singer, U., Polyak, A., Hayes, T., Yin, X., An, J., Zhang, S., Hu, Q., Yang, H., Ashual, O., Gafni, O., et al. Make-a-Video: Text-to-video generation without text-video data. *arXiv:2209.14792*, 2022.

Stepputtis, S., Campbell, J., Phielipp, M., Lee, S., Baral, C., and Ben Amor, H. Language-conditioned imitation learning for robot manipulation tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 13139–13150, 2020.

Stooke, A., Mahajan, A., Barros, C., Deck, C., Bauer, J., Sygnowski, J., Trebacz, M., Jaderberg, M., Mathieu, M., et al. Open-ended learning leads to generally capable agents. *arXiv:2107.12808*, 2021.

Sun, Q., Fang, Y., Wu, L., Wang, X., and Cao, Y. EVA-CLIP: Improved training techniques for CLIP at scale. *arXiv:2303.15389*, 2023.

Tam, A., Rabinowitz, N., Lampinen, A., Roy, N. A., Chan, S., Strouse, D., Wang, J., Banino, A., and Hill, F. Semantic exploration from language abstractions and pretrained representations. In *Advances in Neural Information Processing Systems*, volume 35, pp. 25377–25389, 2022.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. DeepMind Control Suite. *arXiv:1801.00690*, 2018.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IEEE International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Canton-Ferrer, C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models. *arXiv:2307.09288*, 2023.

Tschannen, M., Kumar, M., Steiner, A., Zhai, X., Houlsby, N., and Beyer, L. Image captioners are scalable vision learners too. *arXiv:2306.07915*, 2023.

Wang, M., Xu, X., Yue, Q., and Wang, Y. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *arXiv:2101.12631*, 2021.

Xu, H., Xie, S., Tan, X. E., Huang, P.-Y., Howes, R., Sharma, V., Li, S.-W., Ghosh, G., Zettlemoyer, L., and Feichtenhofer, C. Demystifying CLIP data. *arXiv:2309.16671*, 2023.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pp. 1094–1100, 2020.

Yu, W., Gileadi, N., Fu, C., Kirmani, S., Lee, K.-H., Arenas, M. G., Chiang, H.-T. L., Erez, T., Hasenclever, L., Humplik, J., et al. Language to rewards for robotic skill synthesis. *arXiv:2306.08647*, 2023.

Yuksekgonul, M., Bianchi, F., Kalluri, P., Jurafsky, D., and Zou, J. When and why vision-language models behave like bags-of-words, and what to do about it? In *The Eleventh International Conference on Learning Representations*, 2022.

Zeltner, T., Rousselle, F., Weidlich, A., Clarberg, P., Novák, J., Bitterli, B., Evans, A., Davidovič, T., Kallweit, S., and Lefohn, A. Real-time neural appearance models. *arXiv:2305.02678*, 2023.

Zeng, A., Attarian, M., Ichter, B., Choromanski, K., Wong, A., Welker, S., Tombari, F., Purohit, A., Ryoo, M., Sindhwani, V., et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv:2204.00598*, 2022.

Zhai, X., Mustafa, B., Kolesnikov, A., and Beyer, L. Sigmoid loss for language image pre-training. *arXiv:2303.15343*, 2023.

Zhang, L., Rao, A., and Agrawala, M. Adding conditional control to text-to-image diffusion models. In *IEEE International Conference on Computer Vision*, pp. 3836–3847, 2023.

Zhou, H., Yao, X., Meng, Y., Sun, S., Bing, Z., Huang, K., and Knoll, A. Language-conditioned learning for robotic manipulation: A survey. *arXiv:2312.10807*, 2023.

Zhou, L. and Small, K. Inverse reinforcement learning with natural language goals. In *AAAI Conference on Artificial Intelligence*, volume 35, pp. 11116–11124, May 2021.

A. Environment, Cameras and Configurations

In this section, we first provide details about the Humanoid-plus-cube environment, then we describe the camera settings used to render that environment, and finally we detail each dimension of the environment's configuration space.

A.1. Environment Details

We use the Humanoid environment from OpenAI's Gym framework (Brockman et al., 2016), originally designed for locomotion, which we modify as follows:

- The original Humanoid does not have feet. This is problematic, as we expect the agent to balance in diverse poses. We therefore augment the XML definition of the Humanoid model with feet, borrowed from the Humanoid model in the DeepMind Control Suite (Tassa et al., 2018). This modification adds two dimensions to the action space (for ankle control) and four dimensions to the state space (ankle-foot angles and angular velocities).
- We add a cube to the environment, with sides of length 0.15 m. This allows agent-object interactions, other than agent-floor interactions. This adds 14 dimensions to the state space (7 dimensions for the cube position and orientation and 7 for the corresponding velocities). The cube is modelled as a homogeneous volume with mass 0.5 kg.
- Episodes terminate after 100 steps. We therefore include the episode time-step in the state. To do otherwise would negatively impact policy performance (Pardo et al., 2018).
- We delete some of the original 376 state dimensions, as they are not relevant. Notably, the 84-dimensional vector of external forces is *always* zero when using MuJoCo > 2.0.[3]

In summary, this results in a Humanoid-plus-cube environment, with action space $[-1, 1]^{19}$ and an observation space of 343 dimensions.

Initial state distribution. The original Humanoid environment specifies a *default state*, in which the Humanoid is standing up, facing forward, with all velocities equal to zero. We extend this default state by setting the ankle angles such that the feet are at $6.5°$ to the floor (toes pointing up), setting the Cartesian coordinates of the cube to be $(-0.15, 0.4, 0.15)$, and setting its orientation to be the quarternion $(0, 0, 0, 1)$, so that its sides are parallel to the axes. All corresponding velocities are zero. As in the original Humanoid environment, we sample the initial state of each episode by adding uniform noise on $[-0.01, 0.01]$ independently to all positions and velocities of this default state.

A.2. Camera Settings

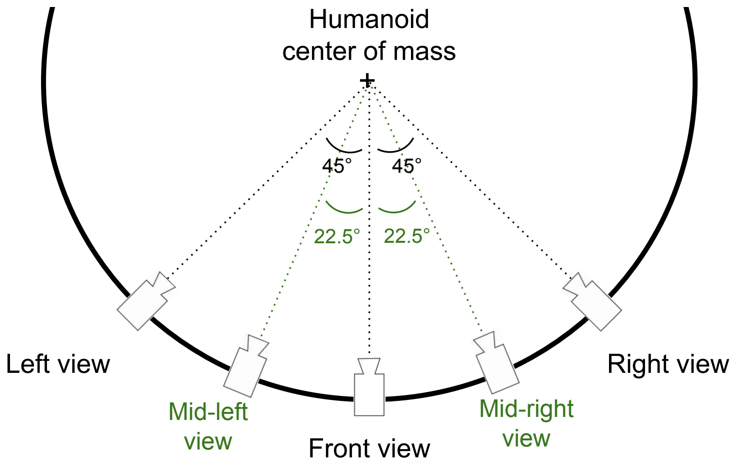


Figure 6. Positions and orientations of the different cameras with respect to the Humanoid center of mass.

[3]This is explained in the environment documentation at `https://www.gymlibrary.dev/environments/mujoco/humanoid/`.

Rendering is performed using MuJoCo rendering functions, which we control by defining cameras in the XML file describing the environment. Our rendering functions share most of their settings: directional lightening with shadow casting, default MuJoCo camera field of view, a wood-colored Humanoid with color `#CC9966`, a grey floor with color `#B8B8B8`, and a blue cube with color `#0033FF`. Each camera generates RGB images of size $224 \times 224 \times 3$, which is the default image resolution of most VLM image encoders. The cameras all point toward and follow the Humanoid's center of mass (COM). They are placed 2.65 m away from the Humanoid COM: this ensures that the cube and the whole body of the Humanoid are visible in the rendered images, for most configurations. The camera heights are initialized 35 centimeters above the Humanoid COM: this ensures that the upper surface of the Humanoid is partly visible, even when the Humanoid is lying on the floor.

As illustrated in Figure 6, the cameras differ in their orientations and positions. The front camera is first placed facing the Humanoid in its default state. The left, mid-left, mid-right and right cameras are equally distant from the Humanoid COM, but are rotated by $-45°$, $-22.5°$, $22.5°$ and $45°$ relative to the front camera respectively, about a vertical axis. The left, right and front views are used to compute the multiview configuration-text score, while the mid-left and mid-right views are only used in the robustness-to-viewpoint evaluations of Section 4.3 and Appendix C.

Clearly, VLM scores depend upon the choice of colours and textures, employed while rendering. There is thus ample room for future research to explore alternative rendering functions, possibly using image-to-image (Zhang et al., 2023) or video-to-video realism enhancement models (Zeltner et al., 2023; Chu et al., 2023) on top of rendered images.

A.3. Configurations

In our modified Humanoid environment, the configuration space $\mathcal{Q}$ has 31 dimensions, as listed in Table 4. These encompass the torso height, the relative positions of its joints, the Cartesian coordinates of the cube, and a quaternion giving the cube's orientation.

Configuration Dimension	**Description**
0	z-coordinate of the torso (centre)
1	x-orientation of the torso (centre)
2	y-orientation of the torso (centre)
3	z-orientation of the torso (centre)
4	w-orientation of the torso (centre)
5	z-angle of the abdomen (in lower waist)
6	y-angle of the abdomen (in lower waist)
7	x-angle of the abdomen (in pelvis)
8	x-coordinate of angle between pelvis and right hip (in right thigh)
9	z-coordinate of angle between pelvis and right hip (in right thigh)
10	y-coordinate of angle between pelvis and right hip (in right thigh)
11	angle between right hip and the right shin (in right knee)
12	x-coordinate of angle between pelvis and left hip (in left thigh)
13	z-coordinate of angle between pelvis and left hip (in left thigh)
14	y-coordinate of angle between pelvis and left hip (in left thigh)
15	angle between left hip and the left shin (in left knee)
16	coordinate-1 (multi-axis) angle between torso and right arm (in right upper arm)
17	coordinate-2 (multi-axis) angle between torso and right arm (in right upper arm)
18	angle between right upper arm and right lower arm (in right elbow)
19	coordinate-1 (multi-axis) angle between torso and left arm (in left upper arm)
20	coordinate-2 (multi-axis) angle between torso and left arm (in left upper arm)
21	angle between left upper arm and left lower arm (in left elbow)
22	angle between left shin and left ankle
23	angle between right shin and right ankle
24	x-coordinate of the cube with respect to the torso x-coordinate
25	y-coordinate of the cube with respect to the torso y-coordinate
26	z-coordinate of the cube
27	x-orientation of the cube
28	y-orientation of the cube
29	z-orientation of the cube
30	w-orientation of the cube

Table 4. Description of each dimension of the configuration space of the Humanoid-plus-cube environment. (The terminology is that of the original Humanoid documentation.)

B. Dataset Sampling and Retrieval

This section gives a detailed explanation of the configuration-dataset sampling methods introduced in Section 3.2.1 of the main paper. Then it discusses the effect of dataset size on the quality of the retrieved configurations, and on the time and memory required for retrieval.

B.1. Random Policy Dataset

This dataset is constructed by collecting all configurations encountered by a random policy. Episodes start in a state sampled from the initial state distribution detailed in Section A, with one exception: we sampled the initial height of the cube uniformly on the interval $[0.15, 1]$, as such configurations are relevant but unlikely to be encountered without this change. The policy then performs actions that are uniformly sampled in the action space at each time step.

B.2. Uniform Sampling Dataset

This dataset is constructed by uniformly sampling the configuration space $\mathcal{Q}$, whose components were described in Table 4. We uniformly and independently sample each component of the configuration within the following bounds:

- Configuration dimension 0 represents the height of the Humanoid's torso, which is not bounded in the Humanoid XML file. Instead, we sample this component from the interval $[0.1, 1.45]$. A height of 0.1 m is the lowest the torso can be without penetrating the floor. If the Humanoid's torso is at a height of 1.45 m, and it is standing up straight, then its feet are 0.2 m above the floor.
- Configuration dimensions 1–4 and 27–30 are quaternions. Therefore we sample them from the interval $[-1, 1]$.
- For configuration dimensions 5–23, which represent relative joint angles, we use the bounds specified in the Humanoid XML file.
- Configuration dimensions 24 and 25 represent the x and y components of the cube's Cartesian coordinates relative to the Humanoid's torso. We sample these from the interval $[-1, 1]$. This ensures that the cube will be visible in most rendered images.
- Configuration dimension 26 is the height of the cube, which is unbounded. We sample this from the interval $[0.145, 1]$ to avoid cube-floor interpenetration.[4]

Sampled configurations may not be admissible, we therefore project them to nearby admissible configurations with $\mathcal{P}_{\mathcal{Q}^{\mathrm{a}}}$ (the operator defined in Section 3.2.2). Because of this projection, the method does not truly result in uniform samples. However, configurations involving contact are obviously important (for instance, standing with feet in contact with the floor, or arms holding a cube) and truly uniform sampling would typically have zero probability of generating such samples.

The uniform sampling procedure generates millions of configurations in under an hour, with most of the time spent projecting configurations. However, computing the multiview embeddings of 2.5×10^6 sampled configurations takes about 58 hours.

B.3. Embedding-Diversity Dataset

The process used to construct the embedding-diversity dataset was as follows:

1. Initialize the dataset with $500\,000$ configurations sampled with the random policy.
2. Compute their exact configuration embeddings (4) using the rendering functions and the VLM.
3. While the dataset has not reached the required size:
   (a) Train a distilled model using the current dataset (as detailed in Appendix D).
   (b) Sample $n = 256\,000$ configurations uniformly without replacement from the current dataset.
   (c) Minimize the maximum pairwise embedding similarity of these configurations by performing $k = 100$ steps of projected gradient descent on the loss given in Section 3.2.1. (We only optimize $n = 256\,000$ per batch, due to the $O(n^2)$ cost of computing this loss.) After each gradient step, project onto nearby admissible configurations using the operator $\mathcal{P}_{\mathcal{Q}^{\mathrm{a}}}$.

[4]Retrospectively, the upper limit of 1 m on the cube height is rather low for some tasks, such as task 121, *"pretending to lift a heavy weight overhead, like a weightlifter"*. However, it is appropriate for most tasks and is not the cause for the inferiority of the uniform sampling method relative to the embedding-diversity method.

(d) Compute the exact configuration embeddings (4) of the optimized configurations.
(e) Include these configurations in the configuration dataset.

This method takes about 64 hours to generate a dataset of 2.5×10^6 configurations with their corresponding multiview configuration embeddings. Even though the process involves repeated training of a distilled model, as well as the optimization of an objective involving many evaluations of the distilled model, most of the run time is spent computing the exact configuration embeddings. This is because computing the exact multiview configuration embeddings is $40\,000\times$ slower than evaluating the distilled model, as discussed in Section 4.5.

B.4. Effect of Dataset Size

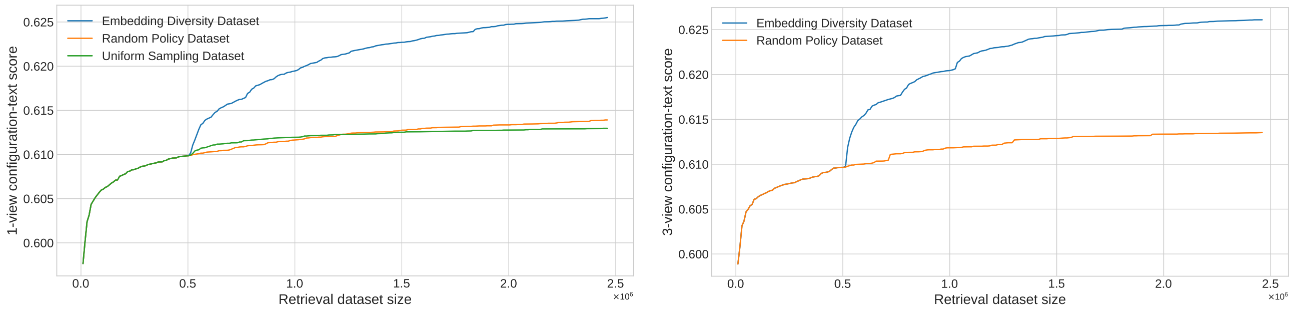


Figure 7. Single-view (left) and multiview (right) configuration-text scores evaluated by EVA02-E-14+ on the best configuration retrieved from each of the three types of dataset, with respect to the dataset size, averaged over the 256 texts. (Results for uniform sampling with multiview scores are not shown due to a lack of time.)

Figure 7 shows the average configuration-text scores, for one- and three-view versions of those scores, as a function of the dataset size. Although the embedding-diversity dataset-generation process extends the dataset in batches of size $500\,000$ or $256\,000$, this plot shows scores for intermediate dataset sizes, generated by randomly sampling an appropriate number of configurations from each new batch. The plots show that an embedding-diversity dataset of size $600\,000$ outperforms a random policy dataset that is $4\times$ larger. The behaviour of the uniform sampling dataset is similar to that of the random policy dataset for single-view scores.

The shape of the curves for the embedding-diversity method can be explained as follows. For datasets smaller than $500\,000$ samples, the embedding diversity samples are all generated by the random policy. Therefore, the curves for random policy and embedding diversity coincide. But at $500\,000$ samples, there is a cusp in the embedding diversity curve, as new samples now result from minimizing pairwise configuration-embedding similarities. There is a similar cusp every $n = 256\,000$ samples thereafter, corresponding to the retraining of the distilled model and the optimization of a new set of configurations, but those cusps are less visible as there is an overall diminishing return on investment.

Retrieval time and memory. The cost of retrieval of the highest-scoring configuration from a configuration dataset, using naïve brute-force evaluation, increases linearly with the size of that dataset, as long as the collection of precomputed embeddings for that dataset remains small enough to fit in the GPU memory. With an NVIDIA RTX A6000 GPU, the time to retrieve from a dataset of size 2.5×10^6 is 15 milliseconds. With 16-bit floating-point embeddings of dimension 1024, this dataset requires 4.77 GB of VRAM, whereas the GPU has 48 GB of VRAM. Were the retrieval time to become a bottleneck, fast nearest neighbour methods could be employed (Wang et al., 2021).

C. Multiview Scores

Table 5 compares configurations retrieved using front-view configuration-text scores with configurations retrieved using multiview configuration-text scores from the embedding-diversity dataset, for the 256 tasks. These two sets of configurations are evaluated with single-view configuration-text scores from five different viewpoints, extending Table 2 of the main paper to include two additional viewpoints: mid-right and right views. As would be expected, the scores for right and mid-right views are very similar to those for left and mid-left views respectively. The scores of configurations retrieved using the front-view tail off more rapidly with viewing angle than those retrieved using multiview. Therefore multiview configuration evaluation leads to configurations that are more robust to changes in the viewpoints used by the rendering functions.

Evaluator Model	Retrieval # Views	Front view	Left view	Right view	Mid-left view	Mid-right view
EVA02-E-14+ (LAION2B)	1	**0.6260**	0.5835	0.5786	0.6050	0.6030
	3	0.6147	**0.6104**	**0.6123**	**0.6138**	**0.6143**
EVA02-E-14 (LAION2B)	1	**0.3496**	0.3164	0.3120	0.3340	0.3330
	3	0.3442	**0.3386**	**0.3423**	**0.3435**	**0.3445**
ViT-H-14 (DFN5B))	1	**0.4102**	0.3816	0.3813	0.3992	0.4006
	3	0.4092	**0.3999**	**0.4043**	**0.4053**	**0.4082**
ViT-H-14 (MetaCLIP))	1	**0.4197**	0.4021	0.3967	0.4131	0.4121
	3	0.4194	**0.4172**	**0.4106**	**0.4202**	**0.4202**
ViT-bigG-14 (DataComp1B)	1	**0.2771**	0.2554	0.2498	0.2661	0.2634
	3	0.2769	**0.2722**	**0.2698**	**0.2734**	**0.2729**

Table 5. Comparison of single-view configuration-text scores of configurations retrieved with EVA02-E-14+ using one or three views, evaluated from different viewpoints with various VLMs.

D. Distilled Model

This section discusses the architecture, training, accuracy and smoothness of the distilled model.

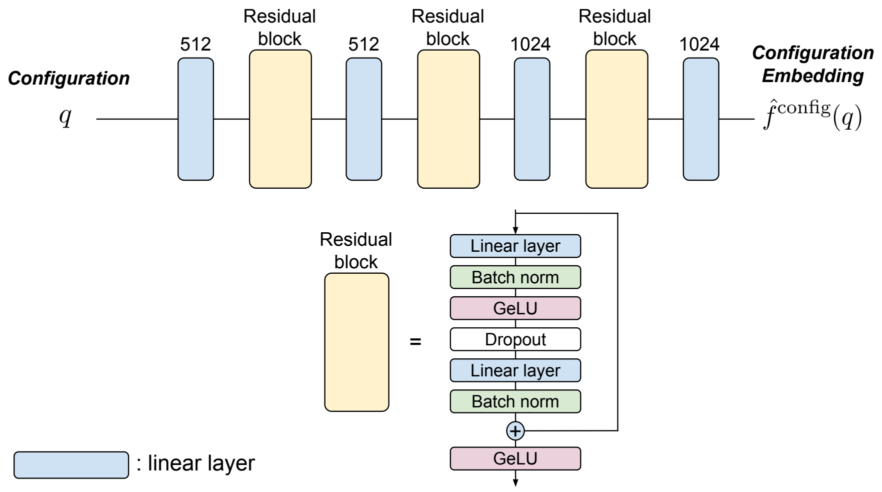


Figure 8. Architecture of the distilled model.

Architecture. Figure 8 illustrates the architecture used for distilled models. This architecture is an MLP, augmented with residual connections and batch normalization inspired by the ResNet architecture (He et al., 2016). It uses GeLU (Hendrycks & Gimpel, 2016) as activation function and has about 5.0×10^6 learnable parameters.

Training. Distilled models are optimized with AdamW (Loshchilov & Hutter, 2017) with a weight decay of 0.01 and a learning rate of 0.001. We use a mean-squared error (MSE) loss on predicted configuration embeddings, with random batches of size 256, and a dropout of 0.01.

Accuracy. To assess accuracy, we train a distilled model on the 2.5×10^6 three-view configuration embeddings of the embedding-diversity dataset. Then we use it to predict the embeddings of the last 2.0×10^6 configurations of the random policy dataset, noting that the datasets have their first 0.5×10^6 configurations in common. We find that the distilled model predicts such configuration embeddings with an RMSE of 4.1×10^{-3} (the normalization used in this RMSE involves division both by the number of components and the number of predictions). The RMSE in configuration-text scores predicted by the distilled model is 9.0×10^{-3}, with the mean taken over the configurations of the random policy dataset and the texts of the 256 tasks. This RMSE is only slightly larger than the amplitude of the 'noisy' oscillations of the VLM score (see Figure 9 and Section H.3).[5]

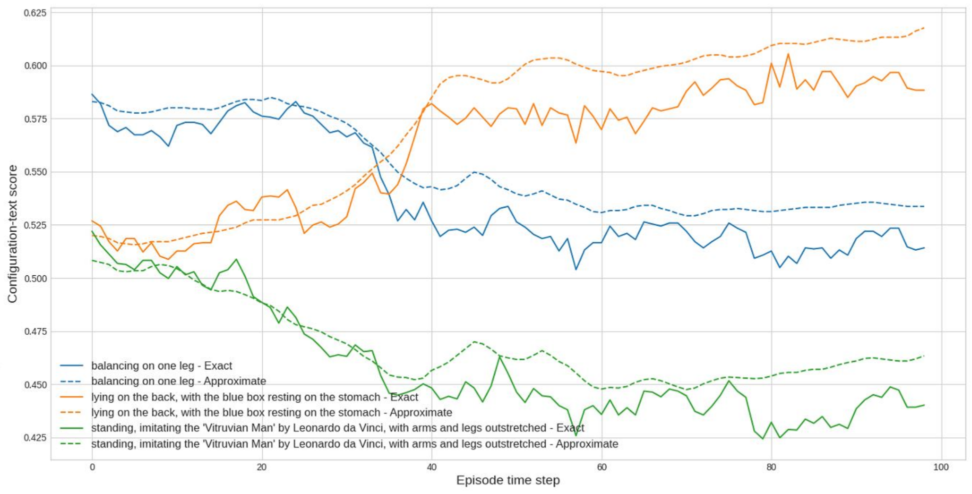


Figure 9. Comparison of exact and approximate configuration-text scores, for a trajectory resulting from the random policy, and for three representative texts.

Smoothness. Figure 9 demonstrates the 'smoothness' of the distilled model relative to the exact configuration-text scores. This plot was generated by rolling out a trajectory with the random policy and evaluating the exact and (distilled-model-based) approximate multiview configuration-text scores of the configurations encountered, using the text embeddings of three representative tasks. We argue that this 'smooth' behaviour of the distilled model makes it more suitable for gradient-based finetuning of configurations than the exact VLM score.

[5]By using an ensemble of distilled models, we were able to slightly reduce these RMSEs, but the reductions were limited by this oscillatory behaviour, so we do not discuss such ensembles any further in this paper.

E. Configuration Finetuning

The process used to finetune a single retrieved configuration is presented in Algorithm 1. In practice, it is applied for a batch of 256 retrieved configurations in parallel, as the configuration-text score is a highly multimodal function of the configuration. First, we precompute the text embedding (Line 2). Then we repeatedly perform gradient ascent, with step-size α, on the configuration-text score approximated with the distilled model (Line 4). Each such gradient step is followed by a projection onto the set of admissible configurations (Line 5), for instance, to ensure that the objects of the environment do not interpenetrate. We repeat this process for 80 finetuning steps.

Algorithm 1 Gradient-based configuration finetuning

```
1: **Input:** Initial configuration $q \in \mathcal{Q}$, text $x \in \mathcal{T}$, VLM text encoder $f^{\text{text}}$, distilled model $\hat{f}^{\text{config}}$ and projection $\mathcal{P}_{\mathcal{Q}^{\text{a}}}$.
2: $z_{\text{t}} \leftarrow f^{\text{text}}(x)$
3: **while** stopping criterion not met **do**
4:     $q \leftarrow q + \alpha \nabla_q (z_{\text{t}} \cdot \hat{f}^{\text{config}}(q))$        {In practice, we use Adam (Kingma & Ba, 2014) for this step.}
5:     $q \leftarrow \mathcal{P}_{\mathcal{Q}^{\text{a}}}(q)$
6: **end while**
7: **return** $q$
```

This choice of the number of finetuning steps is motivated by Figure 10, which shows that the configuration-text score for the best-in-batch configuration plateaus around 60 steps. This figure shows that the plateauing behaviour is observed for both exact and approximate scores, when configurations are selected from the batch using both the exact and approximate scores: there is no sign of overfitting to the distilled model (that is, the exact score does not start decreasing after some number of finetuning iterations).

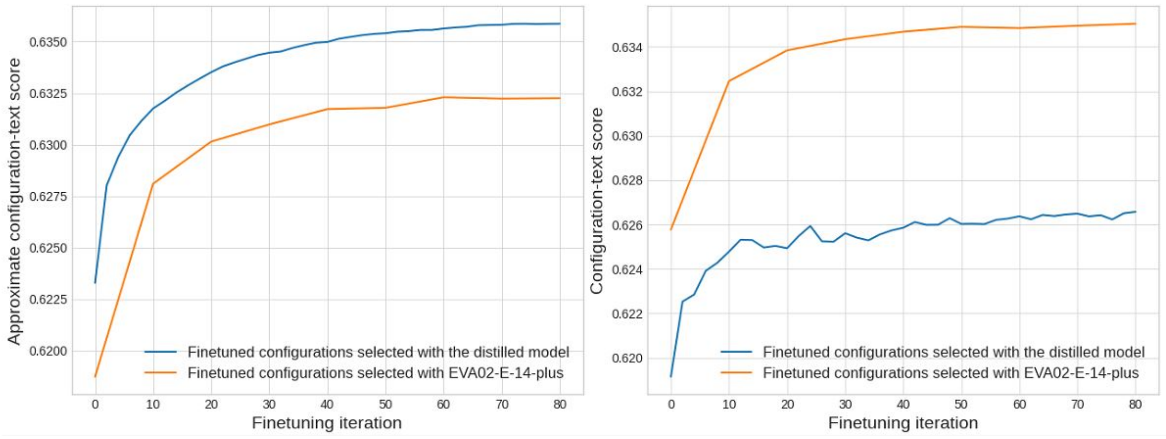


Figure 10. Evolution of the approximate configuration-text score (left) and the exact configuration-text score (right) of configurations selected by the distilled model (blue) and the configurations selected by EVA02-E-14+ (orange) during finetuning, averaged over the 256 tasks. Both scores plateau around 60 finetuning iterations. All scores shown are multiview.

F. LCA Architectures, Hyperparameters and Training

This section discusses the architectures of the LCAs and their training, and illustrates the performance of the MTRL baseline during training.

F.1. LCA Architectures

To ensure a fair comparison between the STRL, MTRL and GCRL agents, they share the same architectures as far as possible. As shown in Figure 11, the MTRL and GCRL agents have a task encoder shared between the policy and the value heads. The agents only differ in their inputs: the MTRL agent receives a text embedding of dimension 1024 as task variable; whereas the GCRL agent receives a target configuration of dimension 31. The STRL agent has neither task variables nor a task encoder. The MTRL, GCRL and STRL agents have about 1.1×10^6, 0.9×10^6 and 0.6×10^6 learnable parameters respectively.

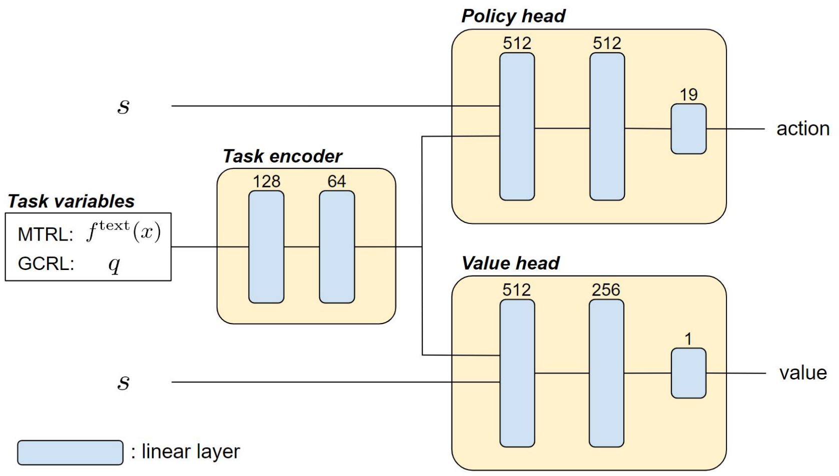


Figure 11. Architecture of MTRL and GCRL agents. Both receive state s as well as task variables as input. The architecture is then composed of three MLPs: (1) the *task encoder*, which embeds the task variables into a vector in $\mathbb{R}^{64}$; (2) the *policy head*, which predicts the action from the concatenation of the state with the embedded task variables; and (3) the *value head*, which predicts the value from the concatenation of the state with the embedded task variables.

F.2. LCA Training

We train the STRL, MTRL and GCRL agents to maximize the finite-horizon discounted sums of their respective rewards (12) and (9), using the proximal policy optimization (PPO) algorithm (Schulman et al., 2017), with the hyperparameters specified in Table 6.

The training times of our methods are as follows, using an NVIDIA RTX A6000 GPU and a 40-core Intel Xeon w7-2475X: we train GCRL for 2×10^9 samples, which takes 149 hours; we train MTRL for 4×10^8 samples, which takes 30 hours; and for each task, we train STRL for 5×10^7 samples, which takes 35 minutes. More training samples are required for the training performance of GCRL to plateau than are required for MTRL. We think this is because the number of goal configurations that GCRL is trained to reach is far greater than the number of tasks that MTRL is trained to perform (2.5×10^6 versus 128), and because the goal configurations of GCRL are diverse.

MTRL and STRL trainings are feasible because we leverage the distilled model: if we had trained the MTRL agent on the exact VLM reward derived from EVA02-E-14+ for the same number of samples, then the reward calculation alone would have taken about 3000 hours (125 days) for the single-view version and about 9000 hours (475 days) for the multiview version. The distilled model reduce these durations by a factor of $40\,000\times$, allowing to spend most training time on data collection and PPO updates.

Figure 12 shows the evolution of the performance of the MTRL agent on both training and test tasks during training. The performance has plateaued, and we do not observe any overfitting (that is, degradation of test-task performance), even after extended training.

Table 6. Hyperparameters used when training the STRL, MTRL and GCRL agents with PPO.

Hyperparameter	**Value**
Clipping	0.2
Discount factor, γ	0.999
GAE parameter, λ	0.95
Update time-step	204 800 (MTRL and GCRL), 25 600 (STRL)
Batch size	102 400 (MTRL and GCRL), 12 800 (STRL)
Epochs	10
Learning rate	5e-4
Learning-rate schedule	linear annealing
Gradient norm clipping	0.5
Value clipping	no
Entropy coefficient	2.5e-2
Value coefficient	0.5
Activation function	GeLU (Hendrycks & Gimpel, 2016)
Optimizer	AdamW (Loshchilov & Hutter, 2017)
Weight decay	0.01

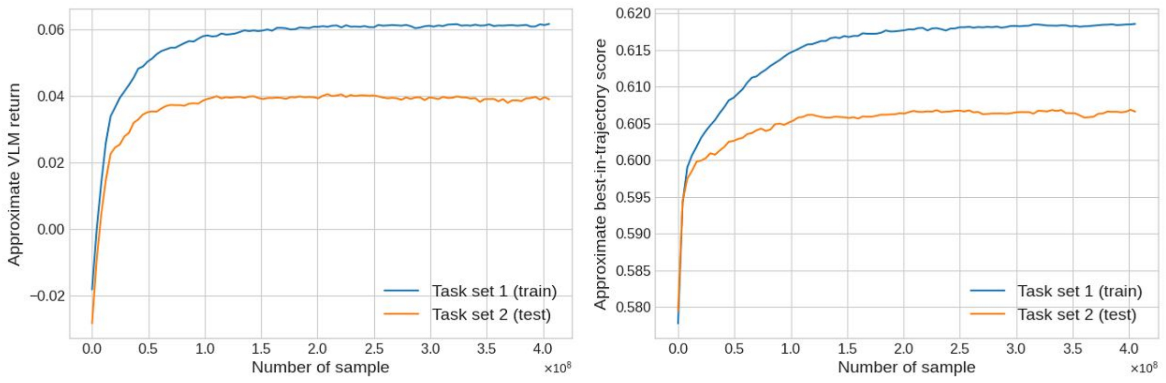


Figure 12. Approximate (multiview) VLM return and approximate best-in-trajectory configuration-text score of the MTRL baseline during training, averaged over the 128 training tasks (blue) and the 128 test tasks (orange), and averaged over 10 episodes per task.

G. Evaluation of LCAs

This section provides additional evaluations of the proposed LCAs. We compare STRL, MTRL and GCRL variants using exact VLM scores, rather than relying on the distilled model (Section G.1); and in terms of the number of tasks where each GCRL variant outperforms the STRL and MTRL baselines (Section G.3). Finally, we present task-by-task performance metrics as polar plots (Section G.4).

G.1. Evaluating LCAs with Exact VLM Scores

Definition. Given trajectory $(s_0, a_0), \ldots, (s_{T-1}, a_{T-1})$, for an episode of length T, the *VLM return* is the discounted sum $\sum_{t=0}^{T-1} \gamma^t R_x(s_t, a_t)$, and the *best-in-trajectory score* is $\max_{t=0,\ldots,T-1} S_{\text{qt}}(\varphi_Q(s_t), x)$, where the VLM reward $R_x(s, a)$ for state s, action a and text x is as defined in equation (11).

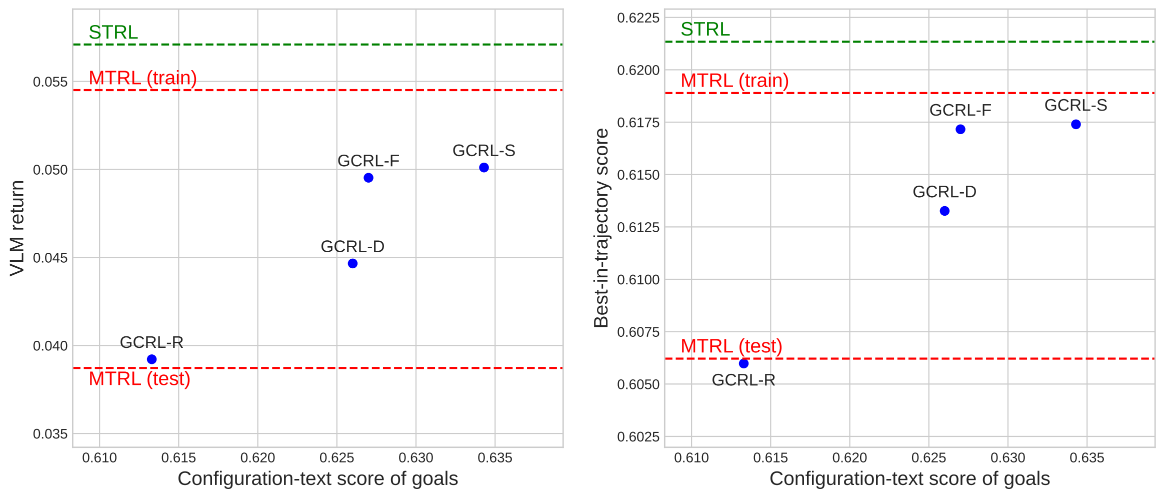


Figure 13. Returns and best-in-trajectory configuration-text scores evaluated by the VLM, for GCRL agents, and for STRL and MTRL baselines. The results are averaged over the 256 tasks.

Results. Figure 13 compares the GCRL, STRL and MTRL agents in terms of their VLM return and best-in-trajectory scores. This figure is thus the analogue of Figure 4, which was based on approximate scores. We make the following observations:

1. GCRL-D, GCRL-F and GCRL-S outperform MTRL (test): thus, decoupling goal-generation and goal-reaching can result in better generalization.
2. Providing goal configurations with higher configuration-text scores to the GCRL agent consistently improves its VLM return and best-in-trajectory score. Indeed, GCRL-F and GCRL-S, which use the highest-quality goals, are competitive with MTRL (train) in terms of best-in-trajectory score.
3. GCRL-S outperforms GCRL-F, in contrast with Figure 4. This is to be expected as GCRL-S uses configurations selected by the VLM, and Figure 13 shows performance evaluated with the VLM, whereas GCRL-F and Figure 4 are both based on approximate scores.
4. STRL outperforms MTRL (train), but only by a small margin relative to the gap between MTRL (test) and MTRL (train).

G.2. LCA Performance by Task Category

We now compare the performance of the LCAs on tasks involving Humanoid-cube interactions with those that do not. To do so, we split the set of 256 tasks (listed in Appendix I) into three groups:

- **Tasks involving cube interaction.** This group consists of tasks that explicitly require Humanoid-cube interactions, such as *"doing a push-up with both feet on the box"*. There are 36 such tasks, with the following IDs: 27, 35, 50, 51, 75, 76, 77, 94, 101, 102, 103, 104, 129, 132, 138, 139, 140, 141, 142, 143, 153, 156, 158, 162, 163, 165, 170, 173, 192, 215, 216, 225, 236, 240, 243 and 244.
- **Tasks potentially involving cube interaction.** This group consists of tasks that could potentially benefit from Humanoid-cube interaction, although they do not explicitly mention such interaction. For instance, when performing the task *"standing on tiptoes, arms reaching up as if trying to touch the ceiling"*, there is some possibility that standing on the cube might help. There are 32 such tasks, with the following IDs: 44, 61, 66, 91, 99, 106, 111, 113, 143, 115, 119, 120, 121, 122, 124, 135, 157, 166, 167, 168, 169, 171, 172, 180, 181, 182, 183, 184, 205, 218, 231 and 233.
- **Tasks involving no cube interaction.** These tasks do not require any form of Humanoid-cube interaction. This last group contains the remaining 188 tasks.

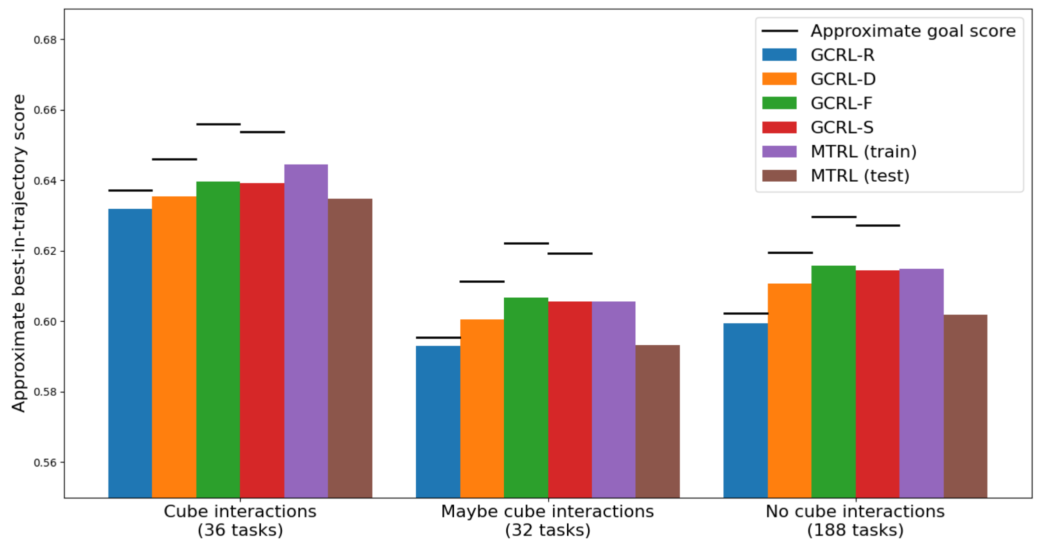


Figure 14. Approximate best-in-trajectory configuration-text scores for GCRL agents and for MTRL baselines averaged over tasks involving Humanoid-cube interactions (left), tasks potentially involving Humanoid-cube interactions (middle) and tasks involving no Humanoid-cube interactions (right). For those LCAs using a goal, the average of the approximate score for the goal is shown as a black line segment.

Figure 14 shows the approximate best-in-trajectory score of each LCA, averaged by task category. For each category, the main result holds: GCRL-D, GCRL-F and GCRL-S outperform MTRL (test) in zero-shot generalization.

Interestingly, MTRL (train) is only better than GCRL-F on tasks requiring Humanoid-cube interactions, attaining a higher approximate best-in-trajectory score on 64% ($\approx 23/36$) of such tasks; and MTRL (train) only beats GCRL-F on 36% ($\approx 80/220$) of the remaining tasks. Upon further investigation, we find that on many tasks requiring Humanoid-cube interaction, neither GCRL-F nor MTRL (train) executes the task in a way that would be satisfactory to a human observer. Part of the problem is that such tasks involve relations between the humanoid and the cube, which are often difficult for VLMs to assess, as discussed in Appendix G.5. In other cases, both MTRL and GCRL appear limited by their exploration: for instance, the VLM can assess if a box has been lifted up, but neither the MTRL nor the GCRL policy can reach that state; whereas STRL, which performs more exploration per task, can successfully lift the cube.

G.3. Comparing Win Rates of LCAs

Table 7 presents the number of tasks where GCRL variants outperform the STRL and MTRL baselines according to a variety of criteria. It shows that providing higher quality goal configurations to the GCRL agent not only increases its average VLM

	GCRL-R	GCRL-D	GCRL-F	GCRL-S
vs. MTRL (test) on exact best-in-trajectory score	120/256	178/256	210/256	210/256
vs. MTRL (test) on exact VLM return	120/256	163/256	198/256	205/256
vs. MTRL (test) on approximate best-in-trajectory score	99/256	185/256	214/256	207/256
vs. MTRL (test) on approximate VLM return	109/256	168/256	203/256	202/256
vs. MTRL (train) on exact best-in-trajectory score	4/256	83/256	124/256	128/256
vs. MTRL (train) on exact VLM return	13/256	64/256	91/256	94/256
vs. MTRL (train) on approximate best-in-trajectory score	6/256	89/256	153/256	130/256
vs. MTRL (train) on approximate VLM return	5/256	59/256	100/256	91/256
vs. STRL on exact best-in-trajectory score	1/256	60/256	91/256	102/256
vs. STRL on exact VLM return	5/256	54/256	70/256	83/256
vs. STRL on approximate best-in-trajectory score	0/256	65/256	118/256	99/256
vs. STRL on approximate VLM return	0/256	37/256	82/256	72/256

Table 7. Win rates of GCRL variants against the MTRL baselines on the MTRL test tasks (top 4 rows) and training tasks (middle 4 rows), and against the STRL baselines (bottom 4 rows). The win rates are given for the best-in-trajectory score and the VLM return, based on both the exact (using the VLM) and approximate (using the distilled model) multiview scores.

return (Figures 4 and 13) but also increases the number of tasks in which it outperforms the baselines. Not only do GCRL-D, -F and -S dominate MTRL (test) for all criteria shown, but also, the best-in-trajectory scores of GCRL-F and -S exceed those of MTRL (train) on at least 48% ($\approx 124/256$) of tasks. Moreover, although STRL was trained to optimize the approximate VLM return, STRL attains a lower (exact and approximate) VLM return than GCRL-F and -S on at least 27% ($\approx 70/256$) of tasks.

The win rates of GCRL-D, -F and -S for best-in-trajectory score, are consistently higher than the corresponding win rates for VLM return. We interpret this phenomenon as follows. Let $(\hat{S}^{(t)}_{\text{qt}})_{t=0}^{T-1}$ be the sequence of (approximate multiview) VLM scores encountered in an episode by some policy. Let score $\hat{S}^{(T)}_{\text{qt}}$ be the *postultimate* score defined as the score for the state predicted to follow the last state of the trajectory, assuming deterministic transitions for simplicity. Then the VLM return for that episode is

$$\sum_{t=0}^{T-1} \gamma^t (\hat{S}^{(t+1)}_{\text{qt}} - \hat{S}^{(t)}_{\text{qt}}) = \gamma^{T-1}\hat{S}^{(T)}_{\text{qt}} - \hat{S}^{(0)}_{\text{qt}} + (1-\gamma)\sum_{t=1}^{T-1} \hat{S}^{(t)}_{\text{qt}} + O\left((1-\gamma)^2\right) \qquad \text{as } \gamma \to 1,$$

where the equality follows from the binomial expansion of $(1-(1-\gamma))^t$. As the MTRL agent attempts to maximize this return, it can be seen as maximizing a weighted sum of the postultimate score $\hat{S}^{(T)}_{\text{qt}}$ and the average score $\sum_{t=0}^{T-1} \hat{S}^{(t)}_{\text{qt}}/T$, noting that the agent has no control over $\hat{S}^{(0)}_{\text{qt}}$. The MTRL policy is free to choose its final configuration as a function of its initial state, and in such a way that its trajectory maximizes this weighted sum. This choice need not result in near-optimal best-in-trajectory scores: maximizing a sum is not generally the same as maximizing the largest term of that sum.

In contrast, the GCRL agent has a fixed goal, and by the binomial expansion argument given above, it can be seen as minimizing a weighted sum of the distance to that goal in its postultimate configuration and the average distance to that goal. As the text-to-goal methods are designed to choose goals that score as highly as possible, if configurations near the goal are reachable and the score is not too rapidly varying near the goal, then by minimizing this weighted sum, the GCRL agent will necessarily attain near-optimal best-in-trajectory scores.

To provide further insight, Figure 15 shows the approximate multiview configuration-text scores of different LCAs as a function of the time step of an episode. The scores of the MTRL agents increase more rapidly than those of the GCRL agents in the first 8 time steps and remain higher until around time step 80. This ensures higher average scores for the MTRL agents on that interval. However, the average scores of GCRL-D, -F and -S increase rapidly in time steps 85–95. One hypothesis for this final burst is that some of the goals are configurations around which it is hard to stabilize. For such goals, the GCRL agent maximizes its trade-off between the distance-to-goal in the postultimate configuration and the average distance-to-goal by waiting in a more-stable-but-not-too-distant configuration until the last few time steps.

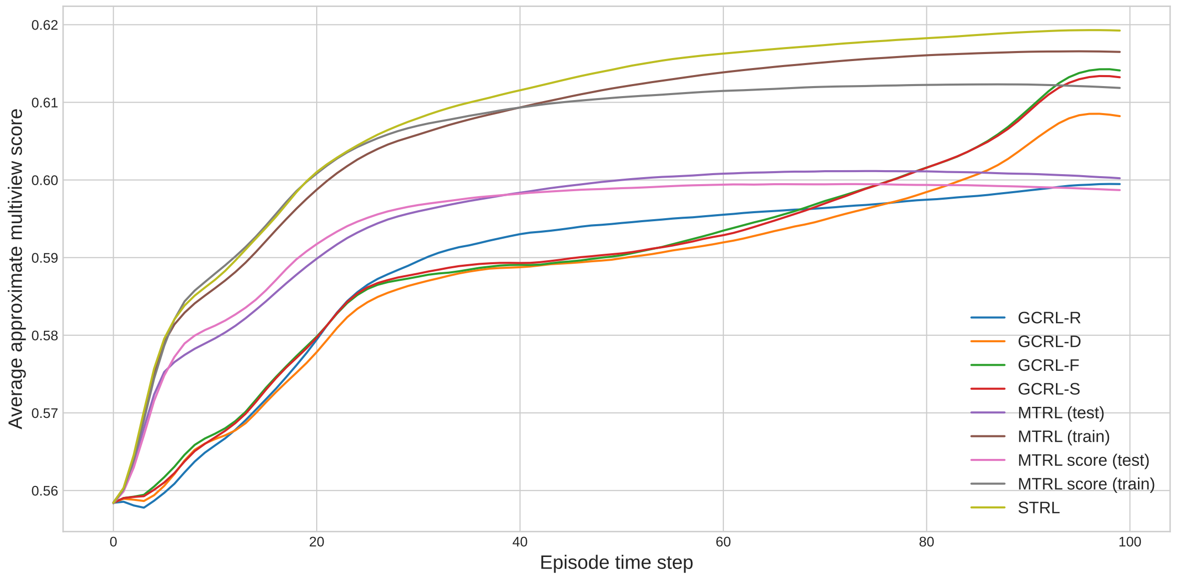


Figure 15. Evolution of the approximate multiview configuration-text score score of GCRL variants and MTRL baselines, averaged over the 256 tasks and 10 episodes per task, with respect to the episode time step.

Time-differenced versus raw VLM rewards. Figure 15 also compares MTRL (train) in brown with *MTRL score (train)* in grey, and MTRL (test) in purple with *MTRL score (test)* in pink. The *MTRL score* variants are trained with the reward

$$\hat{R}_x^{\text{past work}}(s,a) = \hat{S}_{\text{qt}}(\varphi_{\mathcal{Q}}(s), x). \tag{13}$$

This is the reward used by previous works (Mahmoudieh et al., 2022; Fan et al., 2022; Rocamonde et al., 2023; Baumli et al., 2023), except that it is based on multiview configuration-text scores, approximated with our distilled model. It can be seen as an ablated version of our reward function (12), using the score instead of the time difference of scores. We observe that MTRL score (train and test) attain higher scores than MTRL (train and test) respectively from time step 8 to 38. However, MTRL (train and test) dominate after time step 40, and achieve higher peak and final scores. Other authors have used reward (13) to perform tasks defined in terms of the final configuration of the environment, such as the task *'stacking a yellow block on top of a red block'* considered by Mahmoudieh et al. (2022). This result shows that applying our time-differenced reward may be more suitable for such tasks.

G.4. Task-by-Task Evaluation of LCAs

Figure 16 compares the approximate (top) and exact (bottom) task-by-task best-in-trajectory scores of the GCRL-F agent with MTRL (test) and MTRL (train), and with the scores of the finetuned configurations used as goals by GCRL-F. We make the following observations:

- The finetuned goals have the highest approximate configuration-text scores for every task (in the top plot, the orange areas always contain the green, blue and grey magenta areas). The fact that the GCRL agent sometimes fails to reach its goal may be because the goal is intrinsically hard to reach in the time limit of 100 steps, or due to a lack of training.
- While the (exact) best-in-trajectory scores sometimes surpass the goal scores (bottom plot), this happens for $\leq 10\%$ of tasks, and the improvement is always small.
- GCRL-F dominates MTRL (test) for both exact and approximate scores. Examining tasks where MTRL (test) outperforms GCRL-F, we observe a high resemblance between those tasks and the MTRL agent's training tasks. For instance, when the MTRL agent is tested on task 152: *"sitting on the floor with straight legs and touching its head"*, it has been trained on task 151: *"sitting on the floor with straight legs and one arm in the air"*.

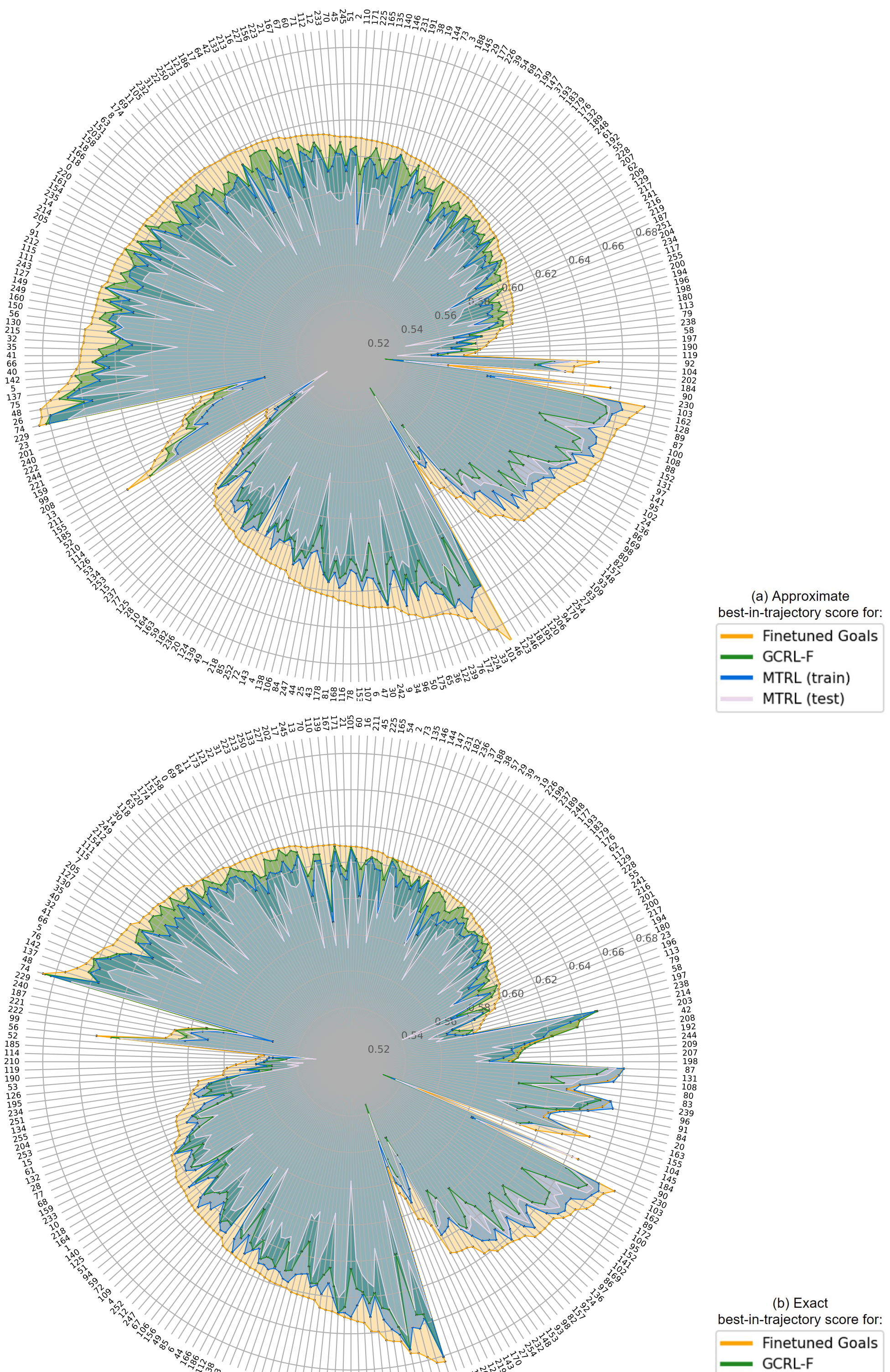


Figure 16. Polar plots of approximate (top) and exact (bottom) multiview configuration-text scores for each task. Task numbers are shown in the outer circle. Scores are shown for: best-in-trajectory configurations of GCRL-F (green), MTRL (train) (blue) and MTRL (test) (pale grey magenta), averaged over 10 episodes per task; and for the finetuned configurations used as goals by GCRL-F (orange). The tasks are grouped by ranking of these scores. For instance, the top region of both plots has orange containing green containing blue containing pale grey magenta, corresponding to the ranking Finetuned Goals $>$ GCRL-F $>$ MTRL (train) $>$ MTRL (test). Within each group, they are sorted by the finetuned goal score.

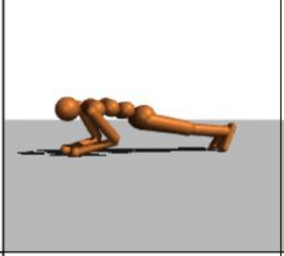
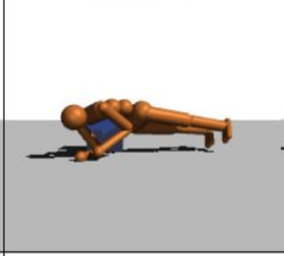
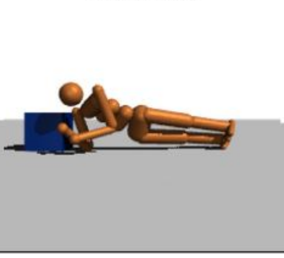
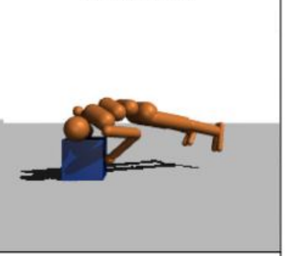
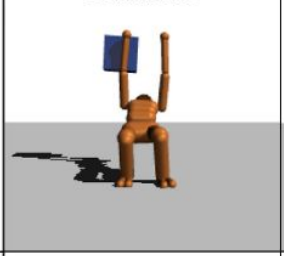
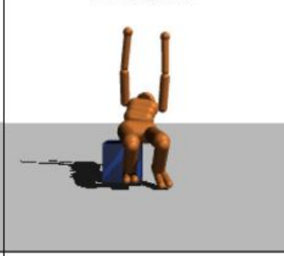
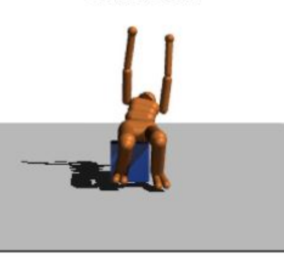
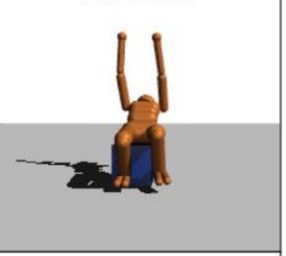
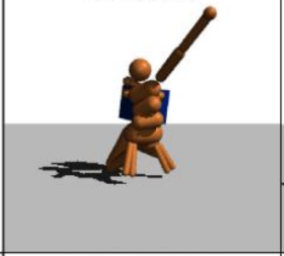
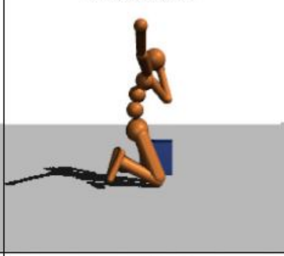
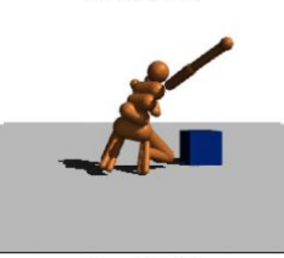
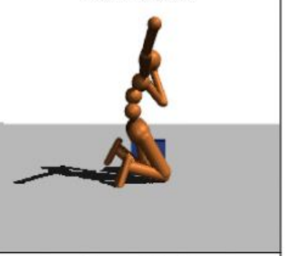
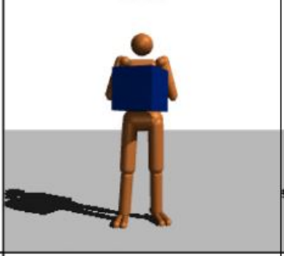
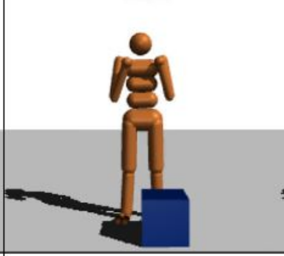
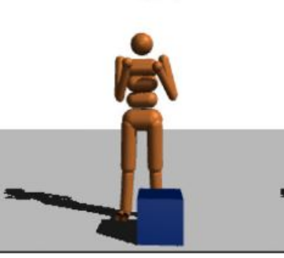
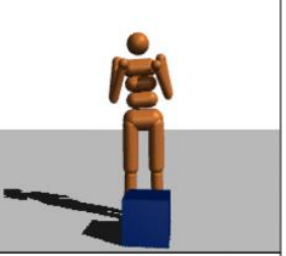
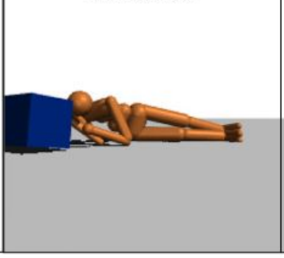
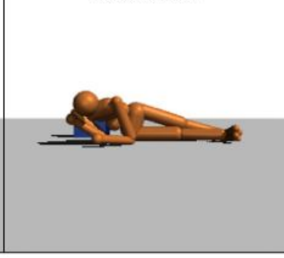
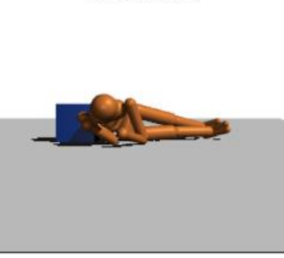
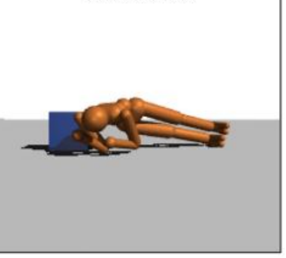
Task	Goal Configuration	Rollout 1	Rollout 2	Rollout 3
ID: 34 *"doing a plank"*	Score: 0.67912 	Score: 0.63076 	Score: 0.62778 	Score: 0.63215 
ID: 121 *"pretending to lift a heavy weight overhead, like a weightlifter"*	Score: 0.60773 	Score: 0.56041 	Score: 0.56057 	Score: 0.56142 
ID: 124 *"pulling an imaginary bow and arrow"*	Score: 0.57807 	Score: 0.52514 	Score: 0.54149 	Score: 0.52720 
ID: 50 *"holding a blue box with its arms"*	Score: 0.62987 	Score: 0.58510 	Score: 0.58492 	Score: 0.58568 
ID: 93 *"lying on its side, with its head propped up by its hand, and with the blue box behind as a backdrop"*	Score: 0.64227 	Score: 0.59183 	Score: 0.60120 	Score: 0.60191 

Figure 17. Front views of the (finetuned) goal configurations and the best-in-trajectory configurations reached by GCRL on its first three rollouts. The tasks shown are the 'worst five failures' of GCRL, in the sense that the difference between the goal score and the best-in-trajectory score (averaged over episodes for that task) is largest.

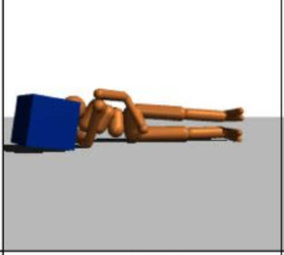
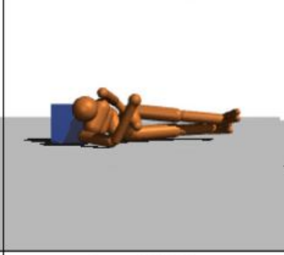
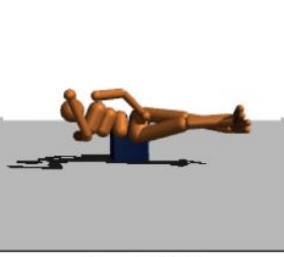
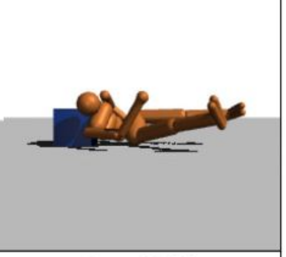
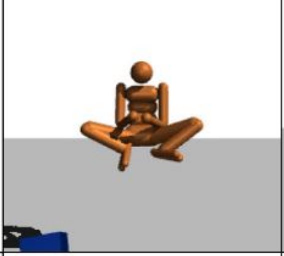
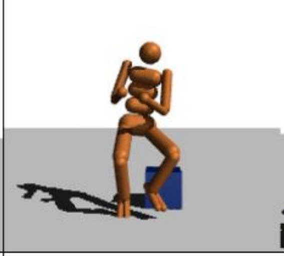
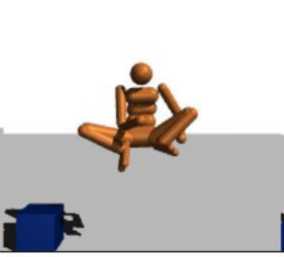
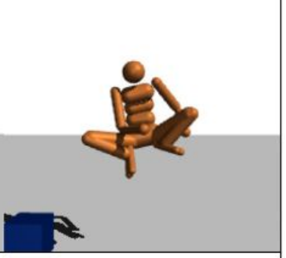
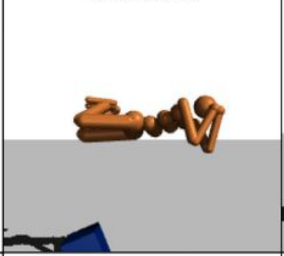
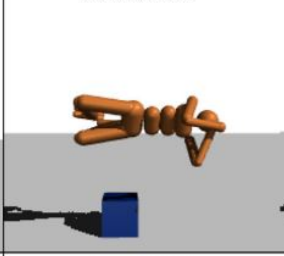
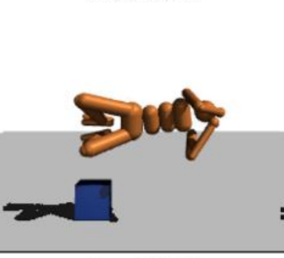
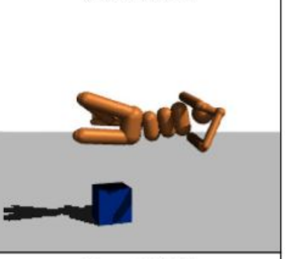
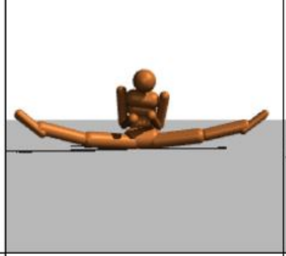
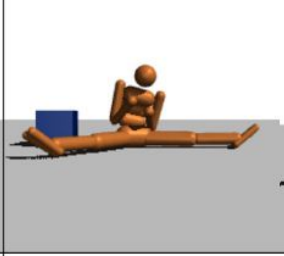
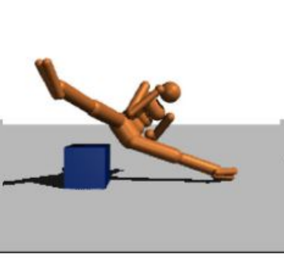
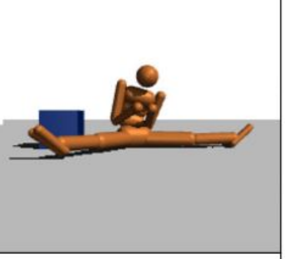
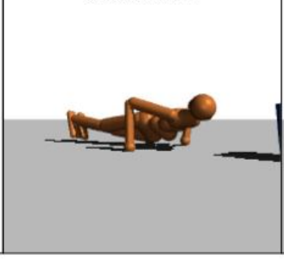
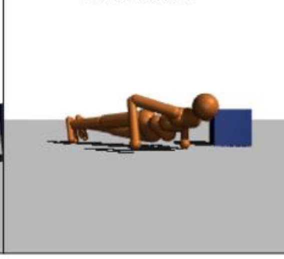
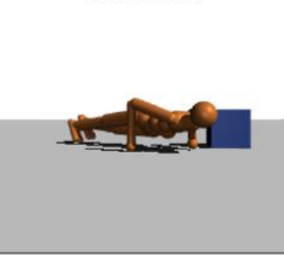
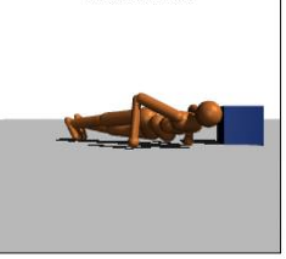
Task	Goal Configuration	Rollout 1	Rollout 2	Rollout 3
ID: 101 *"lying on the back, with arms and legs wrapped around the blue box"*	Score: 0.66179 	Score: 0.61824 	Score: 0.61648 	Score: 0.62021 
ID: 137 *"sitting cross-legged on the floor, hands resting on the knees"*	Score: 0.65076 	Score: 0.58557 	Score: 0.63044 	Score: 0.62494 
ID: 109 *"lying on the stomach, arms folded under the chin, feet dangling in the air"*	Score: 0.66043 	Score: 0.62661 	Score: 0.62508 	Score: 0.62633 
ID: 47 *"doing the front splits"*	Score: 0.69158 	Score: 0.66043 	Score: 0.64878 	Score: 0.65966 
ID: 45 *"doing push-ups"*	Score: 0.64111 	Score: 0.61494 	Score: 0.60960 	Score: 0.61284 

Figure 18. Front views of the (finetuned) goal configurations and the best-in-trajectory configurations reached by GCRL on its first three rollouts. The tasks shown are the 'next worst five failures' of GCRL after those shown in Figure 17. (If the tasks were ranked in order of decreasing difference between the goal score and the average best-in-trajectory score, then these tasks would be placed 6th to 10th).

G.5. Failure Study

We pursue our task-by-task evaluation by analysing failure modes of the GCRL agent. To do so, we investigate the 10 tasks for which the difference between the score of the goal and the average best-in-trajectory score is largest. Figures 17 and 18 illustrate these 10 tasks, showing the goal and the best-in-trajectory configurations reached by GCRL in its first three rollouts. We interpret the failures as follows:

- **Cube still visible (tasks 34, 47 and 45).** The GCRL agent approximately reaches the desired configuration, except that it does not move far enough relative to the cube that the cube goes out of view. Although the cube is irrelevant for these tasks, its presence in the scene results in substantially lower scores. We observe from videos that in general the agent tends to move into a pose and then reposition itself (locomote) relative to the cube. When the agent is upright, it is able to locomote fast. However, in tasks 34, 47 and 45, the agent is lying down or seated and it locomotes by jittering. This mode of locomotion is too slow to travel far enough that the cube goes out of view, in the limited time span of one episode. Perhaps this is a local optimum in policy space, and given more training the agent might learn to locomote then pose.
- **Cube in the way (tasks 93 and 101).** As just remarked, the agent tends to pose then locomote. In these tasks, the consequence is that the Humanoid lies down in front of the cube. From this position it is not able to jitter behind the cube in the limited time. Moreover, the visual difference between being in front of and behind the cube is large, hence the best-in-trajectory score is substantially worse than that of the goal.
- **Horizon problem (tasks 137 and 109).** In the goal configurations of these tasks, the VLM misinterprets the horizon as a structure that the Humanoid can sit or lie on. This leads to unstable goal configurations with the Humanoid floating in the air. Even though the GCRL agent does an impressive job of reaching nearby configurations by jumping, it cannot make such jumps accurately. In particular, attaining a precise apparent alignment between the legs and the horizon seems important to ensure the high score of the goal state. These failures suggest that the proposed text-to-goal method could be further improved by including views from different heights in the multiview score.
- **Cube manipulation (tasks 121, 124 and 50).** In these tasks, the goal configurations require the GCRL agent to manipulate the cube. In the goal configurations of tasks 121 and 124, the VLM seems to interpret the cube as a kettlebell and as a quiver respectively. These goals would require tremendous skill to be reached: in further work, it would be interesting to select goals based on measures of ease of reachability. However, task 50 exposes the fact that the GCRL agent has not yet learned the simple skill of holding a cube with its hands. We suspect this is because few training episodes encompass an increase in the cube's height, so the agent tends to ignore the cube's height. More training and additional goal-reaching training methods such as HER and ASP (Andrychowicz et al., 2017; Plappert et al., 2021) may help. We also plan to investigate longer episodes.

H. Study of VLM Scores

This section studies some properties of VLM scores, and discusses how they impact our text-to-goal methods and the resulting LCAs. All results in this section use the EVA02-E-14+ VLM, which we chose based on its strong classification and retrieval performance (Sun et al., 2023), and based on the arguments of Baumli et al. (2023) and Rocamonde et al. (2023) that larger VLMs engender reward functions that agree better with handcrafted rewards. This VLM is an order of magnitude larger than those employed in previous studies of RL with VLM-based rewards. Its use is only practical thanks to our distilled model, as discussed in Appendix F.2.

We begin by illustrating that VLM scores are highly *multimodal*: they have multiple local maxima when seen as a function of the configuration. Then we discuss situations where VLM scores are *unreliable* and misevaluate configurations. Finally, we show that the score is a highly *oscillatory* function of configuration: the score can increase then decrease rapidly in response to small configuration changes.

H.1. Multimodality of VLM Scores

Figure 19 shows front views of the top-21 configurations retrieved from the embedding-diversity dataset for task 39. The diversity of these configurations demonstrates that the configuration-text score is a highly multimodal function of the configuration. One might provide such a diverse set to a (human or programmatic) user and let them choose which configuration they would like a GCRL agent to aim for, based on criteria other than VLM score alone. For instance, one might choose based on obstacle avoidance and reachability, perhaps using the value function of the GCRL agent, or based on aesthetics or predicted energy consumption. This is another potential advantage of the proposed approach over STRL or MTRL agents, as one has no direct control over which configuration these agents will aim for: one only has indirect control through the text, or through the addition of new terms to the agents' reward to limit energy consumption and so forth.

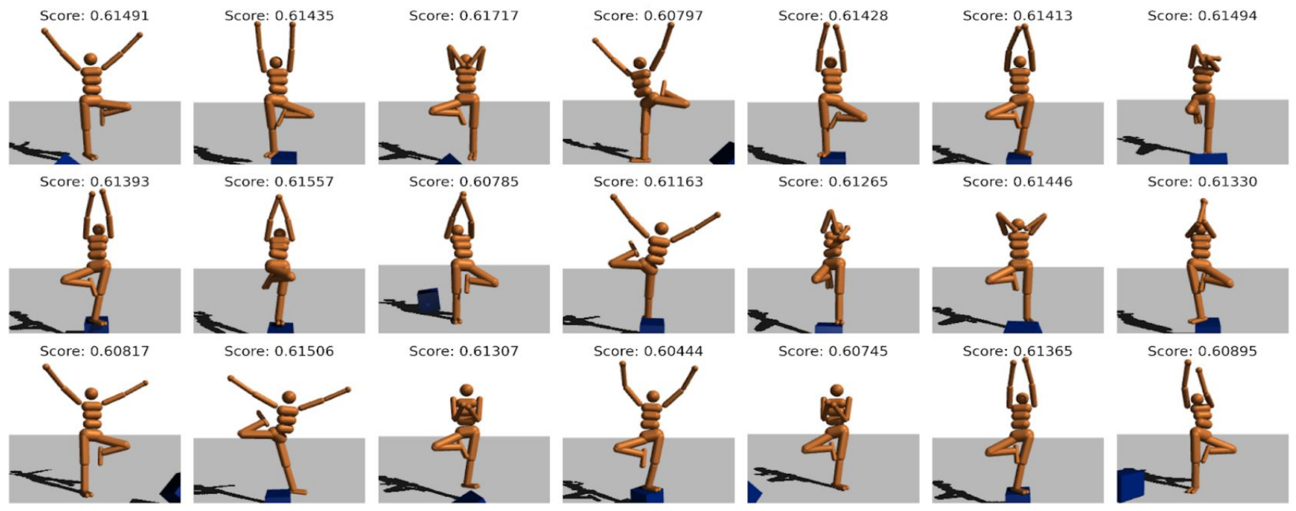


Figure 19. Top 21 configurations retrieved, using multiview scores, from the embedding-diversity dataset for task 39, *"A 3D humanoid model doing a tree yoga pose (Vrkasana)"*.

Figure 20 compares four of the top-21 configurations retrieved for task 39 with images from the internet. The correspondence between these images shows that our text-to-goal method finds diverse configurations that represent naturally occurring variants of a task.

H.2. Unreliability of VLM Scores

Misevaluation. Figure 21 illustrates that VLMs sometimes misevaluate configurations: configurations that might be subjectively preferred by human viewers occasionally receive substantially lower VLM scores than other configurations.

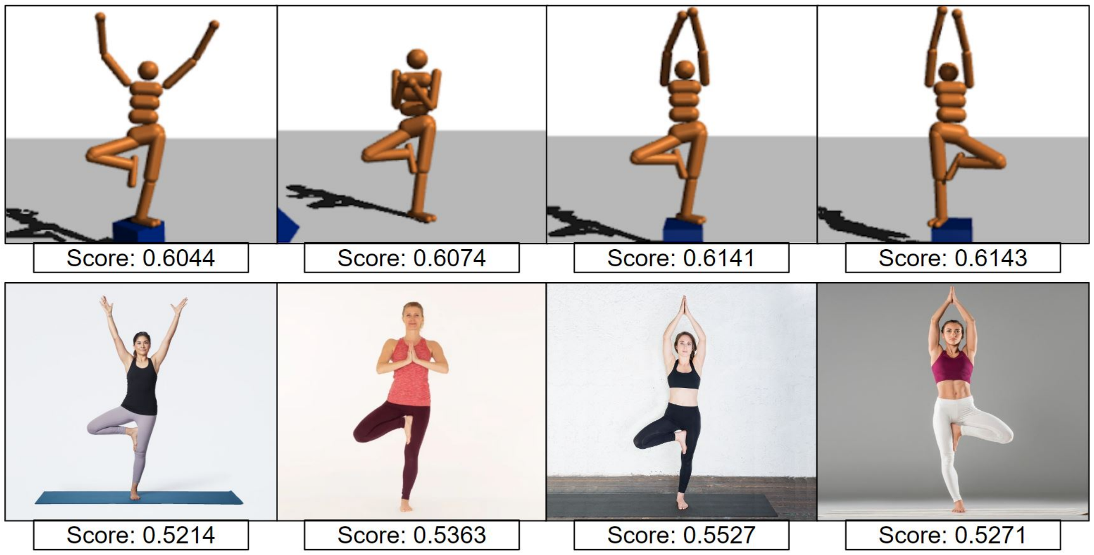


Figure 20. Four of the top-21 retrieved configurations for task 39, *“A 3D humanoid model doing a tree yoga pose (Vrkasana)”* illustrated in Figure 19 matched with images from the internet. The real-world images are evaluated for the text *“A lady doing a tree yoga pose (Vrkasana)”*. Although the Humanoid and real-world images are clear examples of the requested pose, the image-text similarity scores are all less than 0.62, whereas the theoretical maximum of the cosine similarity is 1.

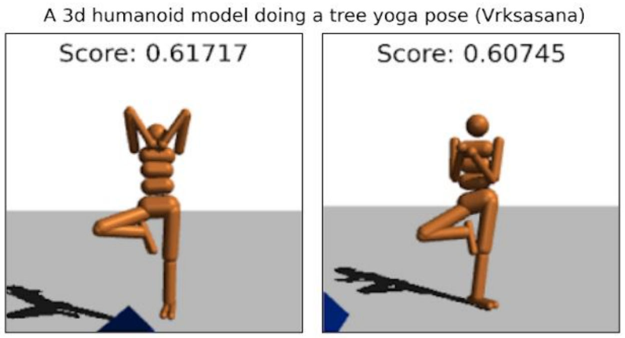


Figure 21. Top-scoring configuration (left) and another top-21 configuration (right) retrieved for task 39, *“A 3D humanoid model doing a tree yoga pose (Vrkasana)”* shown in Figure 19. While the image on the right is often preferred by humans, the VLM gives a significantly higher score to the left image.

Compositionality. Today's VLMs are known to struggle with word order and compositionality (Lewis et al., 2022). To understand this limitation, Yuksekgonul et al. (2022) explore the behaviour of VLMs given shuffled text inputs. They argue that this deficiency stems from the training of VLMs with contrastive losses on retrieval tasks, which provides little incentive for the model to learn to encode order and compositionality cues.

Figure 22 illustrates the failure of VLMs to distinguish between 'left' and 'right'. To generate the left-hand plot, we interpolated configurations linearly between a configuration with the right hand in the air and a configuration with the left hand in the air. As we move through these interpolated configurations, the VLM scores for *"A 3D humanoid model standing up with **left** hand in the air"* and for *"A 3D humanoid model standing up with **right** hand in the air"* are highly correlated. This failure is not due to our choice of rendering functions: we observe the same phenomenon on the right-hand side of Figure 22, which was generated from a 100-frame real-world video. Rather, this failure is due to the VLM's text embedding: the cosine similarity between the text embeddings of these two sentences is 0.9978, thus the VLM scores for these two sentences are almost identical for *any* image embedding.

These failings, along with other known weaknesses such as an inability to count small numbers of objects, limit the diversity of tasks that LCAs derived from VLMs can currently tackle. There have been recent efforts to overcome such limitations, for instance by leveraging reconstruction losses (Tschannen et al., 2023), generating hard negative image-text pairs (Radenovic et al., 2023) and building new benchmarks (Hsieh et al., 2023). Such work promises to enable future LCAs to successfully complete tasks involving compositional relationships between diverse sets of objects.

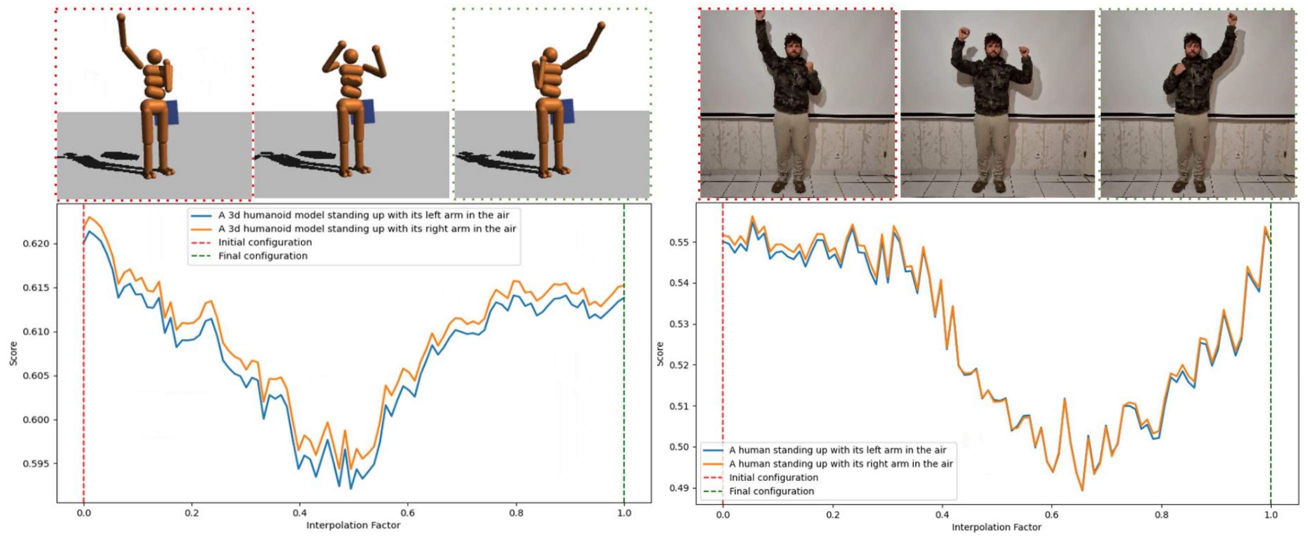


Figure 22. Evolution of VLM score for the text *"A 3D humanoid model standing up with left hand in the air"* and *"A 3D humanoid model standing up with right hand in the air"* on both rendered and real images interpolating between the two described positions.

H.3. Oscillatory Nature of VLM Scores

VLM scores vary rapidly with configuration, exhibiting multiple (aperiodic) oscillations as the configuration varies by a small amount. Figure 23 shows a zoomed-in and more densely sampled version of Figure 22 (left). As the interpolation factor goes from 0.0 to 1.0, the angles made by the links of the Humanoid vary by at most $120°$. Thus, a change in interpolation factor from 0.2400 to 0.2450 corresponds to a change in angle of less than $1°$. Yet, over this small range of interpolation factors, at least 7 local maxima of the VLM score (as a function of the interpolation factor) are visible in the lower part of Figure 23. Therefore, even if the configuration-text score could be differentiated with respect to the configuration, for instance using finite differences or differentiable rendering (Kato et al., 2020), we argue that the derivatives would be of less value for finetuning than the derivatives of a smoother distilled model.

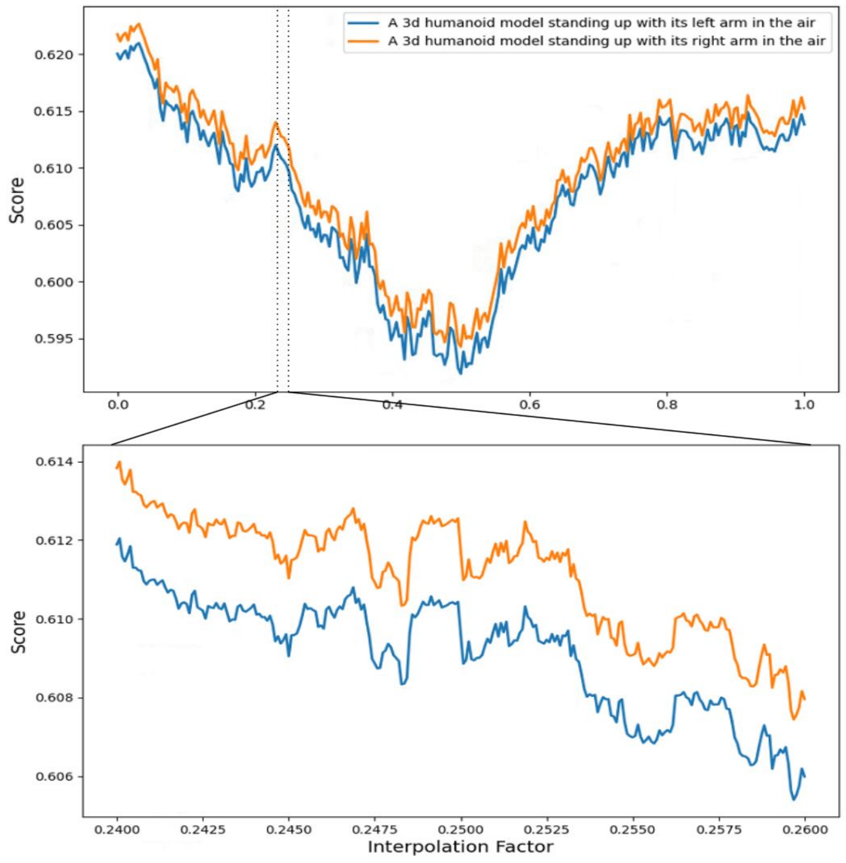


Figure 23. Zoomed-in and more densely sampled version of the left part of Figure 22, illustrating the oscillatory nature of the configuration-text score as a function of configuration.

I. Task Descriptions

This section describes the process used to generate the 256 tasks and then lists those tasks. The tasks are descriptions of desired environment configurations in natural language, which we use to evaluate text-to-goal methods and LCAs. When specifying a task to the agents, we prepend the tasks with *"A 3D humanoid model"*. This choice has been motivated by the remark in the Appendix E.1 of Radford et al. (2021) indicating that prepending *"a photo of"* to the description of each image "boosts CLIP's zero-shot R@1 performance between 1 and 2 points."

Task-generation process. The tasks were generated as follows. We applied ChatGPT (Achiam et al., 2023) with a prompt requesting creative descriptions of a static scene containing a 3D humanoid and a cube, and giving 30 example task descriptions. We deduplicated the resulting tasks and eliminated those texts that referred to objects other than the cube. Finally, we manually edited about 10% of the tasks, as they did not make sense or could have been expressed more naturally, for instance:

- ChatGPT proposed the task *"A 3D humanoid model seated on the floor, one knee raised, chin resting on the knee, echoing the pose of 'The Little Mermaid' statue"*. This does not make sense as the Little Mermaid statue does not have its chin on its knee. Therefore, it was edited to *"A 3D humanoid model seated on the floor, with its knees to the side, echoing the pose of 'The Little Mermaid' statue"*.
- ChatGPT proposed the task *"A 3D humanoid model doing a sidekick stance, like a martial artist"*. A "sidekick" is distinct from a "side kick". Moreover, while there are many named stances in the martial arts such as "horse stance" and "crane stance", we did not identify one named "side kick stance". Therefore, this was edited to *"A 3D humanoid model doing a side kick, like a martial artist"*.

There was no filtering to favour tasks that on which the proposed methods perform well.

List of tasks. The tasks were divided into two sets, with one set used for the training and the other set for the testing of two MTRL agents. The task IDs of these sets are given below, followed by the task descriptions.

Task set 1 IDs: 1, 2, 5, 8, 9, 13, 14, 16, 17, 18, 19, 20, 21, 24, 25, 26, 29, 32, 33, 34, 35, 36, 38, 39, 43, 44, 47, 50, 51, 53, 54, 55, 56, 58, 60, 61, 63, 66, 68, 69, 73, 74, 76, 77, 80, 81, 83, 85, 86, 87, 88, 90, 93, 97, 98, 99, 102, 103, 104, 105, 107, 109, 112, 114, 115, 117, 119, 121, 122, 124, 126, 129, 131, 138, 139, 140, 142, 143, 146, 147, 150, 152, 153, 154, 158, 159, 161, 163, 164, 169, 170, 172, 174, 176, 178, 183, 184, 189, 190, 192, 193, 195, 198, 201, 203, 204, 205, 206, 209, 212, 215, 216, 219, 223, 225, 227, 228, 233, 234, 236, 237, 238, 242, 245, 246, 250, 254, 255.

Task set 2 IDs: 0, 3, 4, 6, 7, 10, 11, 12, 15, 22, 23, 27, 28, 30, 31, 37, 40, 41, 42, 45, 46, 48, 49, 52, 57, 59, 62, 64, 65, 67, 70, 71, 72, 75, 78, 79, 82, 84, 89, 91, 92, 94, 95, 96, 100, 101, 106, 108, 110, 111, 113, 116, 118, 120, 123, 125, 127, 128, 130, 132, 133, 134, 135, 136, 137, 141, 144, 145, 148, 149, 151, 155, 156, 157, 160, 162, 165, 166, 167, 168, 171, 173, 175, 177, 179, 180, 181, 182, 185, 186, 187, 188, 191, 194, 196, 197, 199, 200, 202, 207, 208, 210, 211, 213, 214, 217, 218, 220, 221, 222, 224, 226, 229, 230, 231, 232, 235, 239, 240, 241, 243, 244, 247, 248, 249, 251, 252, 253.

Task 1: *"balancing on one leg"*
Task 2: *"balancing on one leg and waving"*
Task 3: *"balancing on one leg with both hands up"*
Task 4: *"balancing on one leg with folded arms"*
Task 5: *"balancing on one leg with the other leg bent"*
Task 6: *"bowing"*
Task 7: *"crawling"*
Task 8: *"crouching, hands on the floor between feet, head tilted up looking forward"*
Task 9: *"crouching, one hand on the ground for support, looking over the shoulder"*
Task 10: *"crouching, one hand touching the ground, and the other hand shielding its eyes as if looking for something small"*
Task 11: *"curtsying"*
Task 12: *"demonstrating a bridge yoga pose"*
Task 13: *"demonstrating a child yoga pose"*
Task 14: *"demonstrating a cow yoga pose"*
Task 15: *"demonstrating a downward-facing dog yoga pose"*
Task 16: *"demonstrating a happy baby yoga pose"*
Task 17: *"demonstrating a headstand yoga pose"*
Task 18: *"demonstrating a lotus yoga pose"*
Task 19: *"demonstrating a lunge yoga pose"*
Task 20: *"demonstrating a mountain yoga pose"*
Task 21: *"demonstrating a triangle yoga pose"*

Task 22: *“demonstrating a warrior yoga pose”*
Task 23: *“demonstrating a wheel yoga pose”*
Task 24: *“doing a “talk to the hand” pose”*
Task 25: *“doing a ‘victory’ pose (V shape with arms)”*
Task 26: *“doing a bird dog pose”*
Task 27: *“doing a box step-up”*
Task 28: *“doing a camel yoga pose”*
Task 29: *“doing a cobra yoga pose”*
Task 30: *“doing a dab pose”*
Task 31: *“doing a goosestep”*
Task 32: *“doing a half-moon pose”*
Task 33: *“doing a handstand”*
Task 34: *“doing a plank”*
Task 35: *“doing a push-up with both feet on the box”*
Task 36: *“doing a side kick, like a martial artist”*
Task 37: *“doing a side plank”*
Task 38: *“doing a taekwondo back kick”*
Task 39: *“doing a tree yoga pose (Vrksasana)”*
Task 40: *“doing a warrior 3 yoga pose (Virabhadrasana III)”*
Task 41: *“doing an arabesque”*
Task 42: *“doing an arabesque penchée”*
Task 43: *“doing an extended hand-to-big-toe pose”*
Task 44: *“doing crunches”*
Task 45: *“doing push-ups”*
Task 46: *“doing squats”*
Task 47: *“doing the front splits”*
Task 48: *“doing the limbo”*
Task 49: *“doing the side splits”*
Task 50: *“holding a blue box with its arms”*
Task 51: *“holding the blue box so that one corner is in contact with the floor”*
Task 52: *“in a “Saturday Night Fever” dance pose”*
Task 53: *“in a ‘superhero’ stance, hands on hips, chest out, looking confident”*
Task 54: *“in a Bruce Lee fighting stance”*
Task 55: *“in a Namaste pose”*
Task 56: *“in a boxing guard pose”*
Task 57: *“in a crouching position, hands placed on the floor between feet, looking down”*
Task 58: *“in a fencing ‘en garde’ position”*
Task 59: *“in a front double biceps pose”*
Task 60: *“in a front lat spread pose”*
Task 61: *“in a pose reminiscent of Michelangelo’s ‘The Crouching Boy’”*
Task 62: *“in a side chest pose”*
Task 63: *“in a side triceps pose”*
Task 64: *“in a sprint starting pose”*
Task 65: *“in the middle of a ‘moonwalk’ dance step”*
Task 66: *“jumping”*
Task 67: *“kneeling”*
Task 68: *“kneeling and waving”*
Task 69: *“kneeling down and pretending to propose with an imaginary ring”*
Task 70: *“kneeling with both arms in the air”*
Task 71: *“kneeling with both hands up”*
Task 72: *“kneeling with folded arms”*
Task 73: *“kneeling with its hands behind its back”*
Task 74: *“kneeling with its hands on its head”*
Task 75: *“kneeling, one hand on the blue box, looking up as if observing something above”*
Task 76: *“kneeling, one hand on the blue box, the other hand raised above the head, palm facing forward”*
Task 77: *“kneeling, one hand resting on an upright blue box, the other hand on the hip”*
Task 78: *“leaning against an imaginary wall with one arm raised”*
Task 79: *“leaning backward, holding its ankle, in a quad stretch pose”*
Task 80: *“looking for someone”*
Task 81: *“lying down on its side”*
Task 82: *“lying down on the floor”*
Task 83: *“lying down on the floor and waving”*
Task 84: *“lying down on the floor with both hands up”*
Task 85: *“lying down on the floor with folded arms”*
Task 86: *“lying down on the floor with hands on its head”*
Task 87: *“lying face down with arms and legs stretched out, like a ‘starfish’”*
Task 88: *“lying face up, hands under the head as a pillow, one leg crossed over the other”*
Task 89: *“lying flat on the back, both legs raised straight up”*
Task 90: *“lying flat on the back, one leg raised straight up”*
Task 91: *“lying on its back with knees bent, in a sit-up position”*
Task 92: *“lying on its back with one knee bent”*
Task 93: *“lying on its back, forming an ‘X’ shape with spread arms and legs”*
Task 94: *“lying on its side, with its head propped up by its hand, and with the blue box behind as a backdrop”*
Task 95: *“lying on its stomach, propping up its head with its hands”*
Task 96: *“lying on the back, arms and legs slightly spread apart, in a relaxed pose”*
Task 97: *“lying on the back, hands under the head, one leg crossed over the other, ankle on knee”*
Task 98: *“lying on the back, imitating a snow angel movement, arms and legs moving outward and inward”*
Task 99: *“lying on the back, in a ‘fainting’ pose reminiscent of Victorian paintings”*
Task 100: *“lying on the back, one hand on the forehead, the other resting on the stomach”*
Task 101: *“lying on the back, with arms and legs wrapped around the blue box”*

Task 102: *“lying on the back, with the blue box placed under the head like a pillow”*
Task 103: *“lying on the back, with the blue box resting on the stomach”*
Task 104: *“lying on the side, hugging the blue box, with one leg bent”*
Task 105: *“lying on the side, propped up on one elbow”*
Task 106: *“lying on the side, propped up on one elbow, legs crossed at the ankles, in a classic ‘Reclining Venus’ pose”*
Task 107: *“lying on the side, propping head up with one hand, the other arm resting along the body”*
Task 108: *“lying on the stomach, arms and legs outstretched, face turned to one side”*
Task 109: *“lying on the stomach, arms folded under the chin, feet dangling in the air”*
Task 110: *“lying on the stomach, chin resting on hands, legs crossed at the ankles behind”*
Task 111: *“meditating”*
Task 112: *“mimicking the “Karate Kid” crane kick pose”*
Task 113: *“mimicking the “Rodin’s The Thinker” pose”*
Task 114: *“mimicking the iconic ‘Statue of Liberty’ pose”*
Task 115: *“mimicking the pose of Leonardo da Vinci’s ‘The Vitruvian Man’”*
Task 116: *“on all fours, one arm extended forward, the opposite leg extended back”*
Task 117: *“on one knee, other leg bent with foot flat on the floor, and hands on the raised knee”*
Task 118: *“playing an imaginary violin”*
Task 119: *“praying”*
Task 120: *“pretending to be a conductor leading an orchestra”*
Task 121: *“pretending to lift a heavy weight overhead, like a weightlifter”*
Task 122: *“pretending to shoot a basketball”*
Task 123: *“prostrating itself”*
Task 124: *“pulling an imaginary bow and arrow”*
Task 125: *“running”*
Task 126: *“running in the style of Naruto ninjas”*
Task 127: *“saluting like a soldier”*
Task 128: *“seated on the floor, bending forward towards the feet, arms extended forward”*
Task 129: *“seated on the floor, one hand resting on the blue box, looking thoughtful”*
Task 130: *“seated on the floor, with its knees to the side, echoing the pose of ‘The Little Mermaid’ statue”*
Task 131: *“seated on the ground, leaning sideways with one hand supporting the body”*
Task 132: *“seated, leaning against the blue box, legs crossed, one hand supporting the head”*
Task 133: *“seated, leaning forward with its elbows on its knees and its chin resting on its hands”*
Task 134: *“seated, legs stretched out, leaning back on its hands, gazing upward”*
Task 135: *“singing”*
Task 136: *“sitting back on heels, hands resting on thighs”*
Task 137: *“sitting cross-legged on the floor, hands resting on the knees”*
Task 138: *“sitting cross-legged on the floor, one hand on the blue box, the other hand gesturing in mid-air”*
Task 139: *“sitting on the blue box, hands on knees, looking straight ahead”*
Task 140: *“sitting on the blue box, leaning forward with its chin resting on hands”*
Task 141: *“sitting on the blue box, leaning forward, in a pose reminiscent of ‘The Old Guitarist’ by Picasso”*
Task 142: *“sitting on the blue box, legs crossed, one hand on the knee, the other resting on the box”*
Task 143: *“sitting on the blue box, legs dangling”*
Task 144: *“sitting on the floor”*
Task 145: *“sitting on the floor and waving”*
Task 146: *“sitting on the floor in a mermaid pose”*
Task 147: *“sitting on the floor with both hands up”*
Task 148: *“sitting on the floor with folded arms”*
Task 149: *“sitting on the floor with straight legs”*
Task 150: *“sitting on the floor with straight legs and both arms in the air”*
Task 151: *“sitting on the floor with straight legs and one arm in the air”*
Task 152: *“sitting on the floor with straight legs and touching its head”*
Task 153: *“sitting on the floor with the blue box in front, arms resting on the box”*
Task 154: *“sitting on the floor, back straight, with its hands resting on its knees, as if meditating”*
Task 155: *“sitting on the floor, body twisted to one side, looking over the shoulder”*
Task 156: *“sitting on the floor, hugging the blue box to the chest”*
Task 157: *“sitting on the floor, imitating Rodin’s ‘The Thinker’ pose, with chin resting on hand”*
Task 158: *“sitting on the floor, leaning against the blue box with arms crossed”*
Task 159: *“sitting on the floor, leaning back on hands with legs stretched out front”*
Task 160: *“sitting on the floor, leaning on one hand, legs casually stretched out, in a relaxed Bohemian style”*
Task 161: *“sitting on the floor, one arm resting on the raised knee, the other hand supporting the body behind”*
Task 162: *“sitting on the floor, one hand on the blue box, the other raised as if it wished to ask a teacher a question”*
Task 163: *“sitting on the ground, back against the blue box, legs stretched out”*
Task 164: *“sitting on the ground, leaning back on its hands, with its legs bent, in a relaxed ‘La Dolce Vita’ style”*
Task 165: *“sitting on top of a box”*
Task 166: *“sitting with legs crossed and one hand on the chin”*
Task 167: *“sitting with legs crossed, one hand under the chin, and a pensive look, emulating ‘The Thinker’”*
Task 168: *“sitting with legs tucked under, resting hands on knees”*
Task 169: *“sitting with legs wide apart, leaning forward with arms extended between legs”*
Task 170: *“sitting with the blue box placed on its lap, hands resting on the box”*
Task 171: *“sitting, knees drawn up, chin resting on its knees, arms wrapped around its legs”*
Task 172: *“sitting, leaning forward with its elbows on its knees, gazing downward thoughtfully”*
Task 173: *“squatting next to the blue box, resting one elbow on the box”*
Task 174: *“squatting with arms extended forward”*
Task 175: *“squatting, arms wrapped around the knees, head bowed”*
Task 176: *“squatting, with its hands touching the ground, in a sprinter’s starting position”*
Task 177: *“standing and gazing upwards, hands clasped behind the back”*
Task 178: *“standing in a ‘goalkeeper’ stance, arms outstretched to the sides, legs slightly bent”*
Task 179: *“standing in a T-pose (arms extended horizontally)”*
Task 180: *“standing in a contrapposto pose, weight shifted onto one leg, reminiscent of classical Greek statues”*
Task 181: *“standing in a surfing stance, pretending to ride a wave”*

Task 182: *“standing in the ‘Atlas’ pose, as if holding an invisible globe on its shoulders”*
Task 183: *“standing in the ‘Discobolus’ (Discus Thrower) pose, as if ready to throw an imaginary discus”*
Task 184: *“standing knock kneed and pigeon toed”*
Task 185: *“standing like a captain, with hands akimbo (on the hips)”*
Task 186: *“standing like a mime artist pulling an imaginary rope”*
Task 187: *“standing on tiptoes, arms reaching up as if trying to touch the ceiling”*
Task 188: *“standing up”*
Task 189: *“standing up and reaching down to touch the toes”*
Task 190: *“standing up and waving”*
Task 191: *“standing up in Usain Bolt’s celebration pose”*
Task 192: *“standing up on top of the blue box”*
Task 193: *“standing up with bent knees”*
Task 194: *“standing up with both hands up”*
Task 195: *“standing up with folded arms”*
Task 196: *“standing up with hands behind its back”*
Task 197: *“standing up with its hands on its head”*
Task 198: *“standing up with its hands on its hips”*
Task 199: *“standing up with its hands touching above its head”*
Task 200: *“standing up with one hand in the air and one hand on its head”*
Task 201: *“standing up with one hand on its hip and one hand on its head”*
Task 202: *“standing with arms crossed over the chest”*
Task 203: *“standing with arms extended upward, forming a ‘Y’ shape”*
Task 204: *“standing with arms folded behind the head, elbows out, in a relaxed stance”*
Task 205: *“standing with arms outstretched, mimicking the Christ the Redeemer statue in Rio de Janeiro”*
Task 206: *“standing with arms raised high, one foot forward, in a ballet ‘Arabesque’ position”*
Task 207: *“standing with arms wide open, as if ready to give a hug”*
Task 208: *“standing with back arched and hands reaching towards the sky”*
Task 209: *“standing with feet apart, hands clasped together in front at waist level”*
Task 210: *“standing with hands together in front, as if holding an invisible object delicately”*
Task 211: *“standing with one arm bent, fist near the cheek, emulating the iconic ‘Rosie the Riveter’ pose”*
Task 212: *“standing with one arm extended horizontally, pointing to the side”*
Task 213: *“standing with one arm extended outward as if signaling ‘stop’ ”*
Task 214: *“standing with one arm extended upwards, pointing”*
Task 215: *“standing with one foot on the blue box, hands on hips, looking down at the box”*
Task 216: *“standing with one hand extended as if presenting the blue box”*
Task 217: *“standing with one hand on the chest and the other hand raised in a ‘stop’ gesture”*
Task 218: *“standing with one hand on the chin, in a thoughtful pose”*
Task 219: *“standing with one leg crossed over the other, arms hanging loosely at the sides”*
Task 220: *“standing, arms akimbo, with a tilted head, emulating the iconic ‘Superman’ stance”*
Task 221: *“standing, arms interlocked above the head, stretching upward”*
Task 222: *“standing, arms outstretched to the sides, palms facing forward, in a welcoming stance”*
Task 223: *“standing, arms reaching out to the sides at shoulder height, palms up”*
Task 224: *“standing, bending forward at the waist, hands reaching towards the feet”*
Task 225: *“standing, bending one knee, foot on the blue box, mimicking a stretching pose”*
Task 226: *“standing, bending slightly forward with hands on its lower back”*
Task 227: *“standing, bending slightly to one side, one hand on the hip, the other hand touching the side of the head”*
Task 228: *“standing, body twisted, one arm reaching across to touch the opposite shoulder”*
Task 229: *“standing, head tilted back, hands clasped behind the neck”*
Task 230: *“standing, head tilted to one side, hands forming a heart shape at chest level”*
Task 231: *“standing, imitating the ‘Vitruvian Man’ by Leonardo da Vinci, with arms and legs outstretched”*
Task 232: *“standing, leaning slightly forward, as if peering into the distance”*
Task 233: *“standing, mimicking the act of holding a large object above the head with both hands”*
Task 234: *“standing, mimicking the act of looking through an imaginary telescope”*
Task 235: *“standing, one arm bent with hand on waist, the other arm extended, palm up”*
Task 236: *“standing, one arm draped over the blue box, the other hand touching the chin, in a contemplative pose”*
Task 237: *“standing, one arm raised, hand shading the eyes as if looking into the distance”*
Task 238: *“standing, one foot slightly forward, hands clasped at waist level, looking serene”*
Task 239: *“standing, one hand on heart, the other extended outward in a welcoming gesture”*
Task 240: *“standing, one hand on the blue box, as if leaning on it for support”*
Task 241: *“standing, one hand touching the chin, the other arm hanging by the side”*
Task 242: *“standing, one hand touching the temple as if deep in thought”*
Task 243: *“standing, one leg raised and resting on the blue box, hands on hips, looking at the raised foot”*
Task 244: *“standing, one leg slightly raised, foot resting on the blue box, hands on hips”*
Task 245: *“standing, tilting the head to one side, hands relaxed by the sides”*
Task 246: *“stretching its glutes on the floor”*
Task 247: *“stretching its glutes while standing up”*
Task 248: *“stretching its quads on the floor”*
Task 249: *“stretching its quads while standing up”*
Task 250: *“throwing a high kick”*
Task 251: *“throwing an uppercut”*
Task 252: *“touching its knees with its hands”*
Task 253: *“walking”*
Task 254: *“walking like an Egyptian”*
Task 255: *“walking with both arms in the air”*
Task 256: *“walking with both hands touching its head”*

J. Generated and Reached Configurations

This section provides frontal views of the configurations generated by our text-to-goal methods and then the configurations reached by our LCAs and the baseline LCAs for each of the 256 tasks listed in Appendix I.

J.1. Generated Goal Configurations

The following figures show frontal views of the highest-scoring configurations for each of the proposed text-to-goal methods for both single and multiview scores based on the EVA02-E-14+ VLM.

- Figures 24 and 25 show configurations retrieved from the random-policy dataset, for single-view and multiview scores respectively.
- Figures 26 and 27 show configurations retrieved from the embedding-diversity dataset, for single-view and multiview scores respectively.
- Figures 28 and 29 show configurations retrieved from the embedding-diversity dataset and finetuned with the distilled model, for single-view and multiview scores respectively.
- Figures 30 and 31 show configurations retrieved from the embedding-diversity dataset, finetuned with the distilled model, then selected with the VLM, for single-view and multiview scores respectively.

J.2. Reached Configurations

The following figures show frontal views of the best-in-trajectory configurations reached by our LCAs and by the MTRL baselines during a single rollout, based on the multiview text-configuration score.

- Figures 32 show the best-in-trajectory configurations reached by GCRL-R.
- Figures 33 show the best-in-trajectory configurations reached by GCRL-D.
- Figures 34 show the best-in-trajectory configurations reached by GCRL-F.
- Figures 35 show the best-in-trajectory configurations reached by GCRL-S.
- Figures 36 show the best-in-trajectory configurations reached by MTRL evaluated on its training tasks.
- Figures 37 show the best-in-trajectory configurations reached by MTRL evaluated on its test tasks.

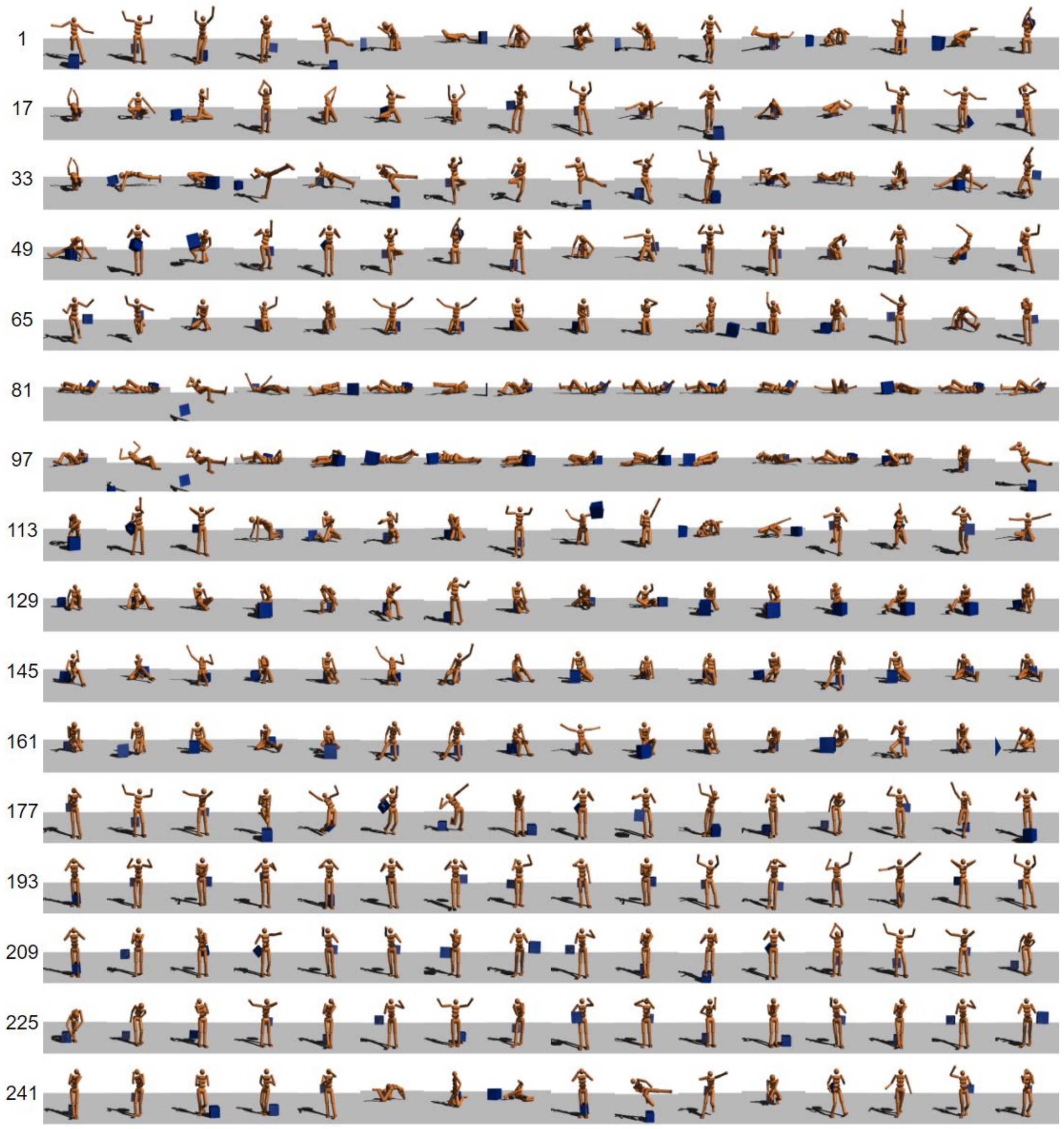


Figure 24. Front views of the highest-scoring configurations retrieved from the random-policy dataset with EVA02-E-14+, using single-view evaluations.

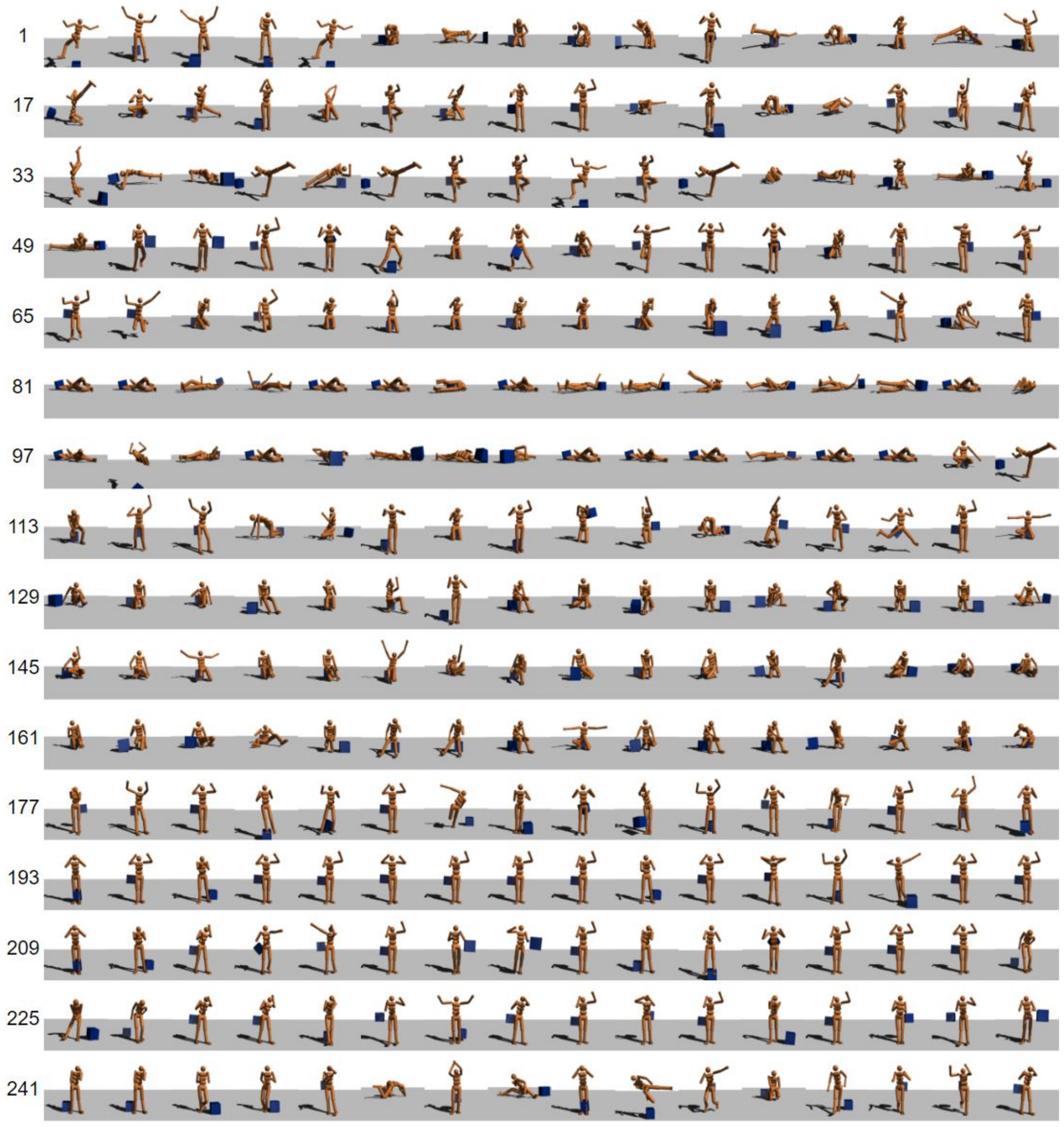


Figure 25. Front views of the highest-scoring configurations retrieved from the random-policy dataset with EVA02-E-14+, using multiview evaluations.

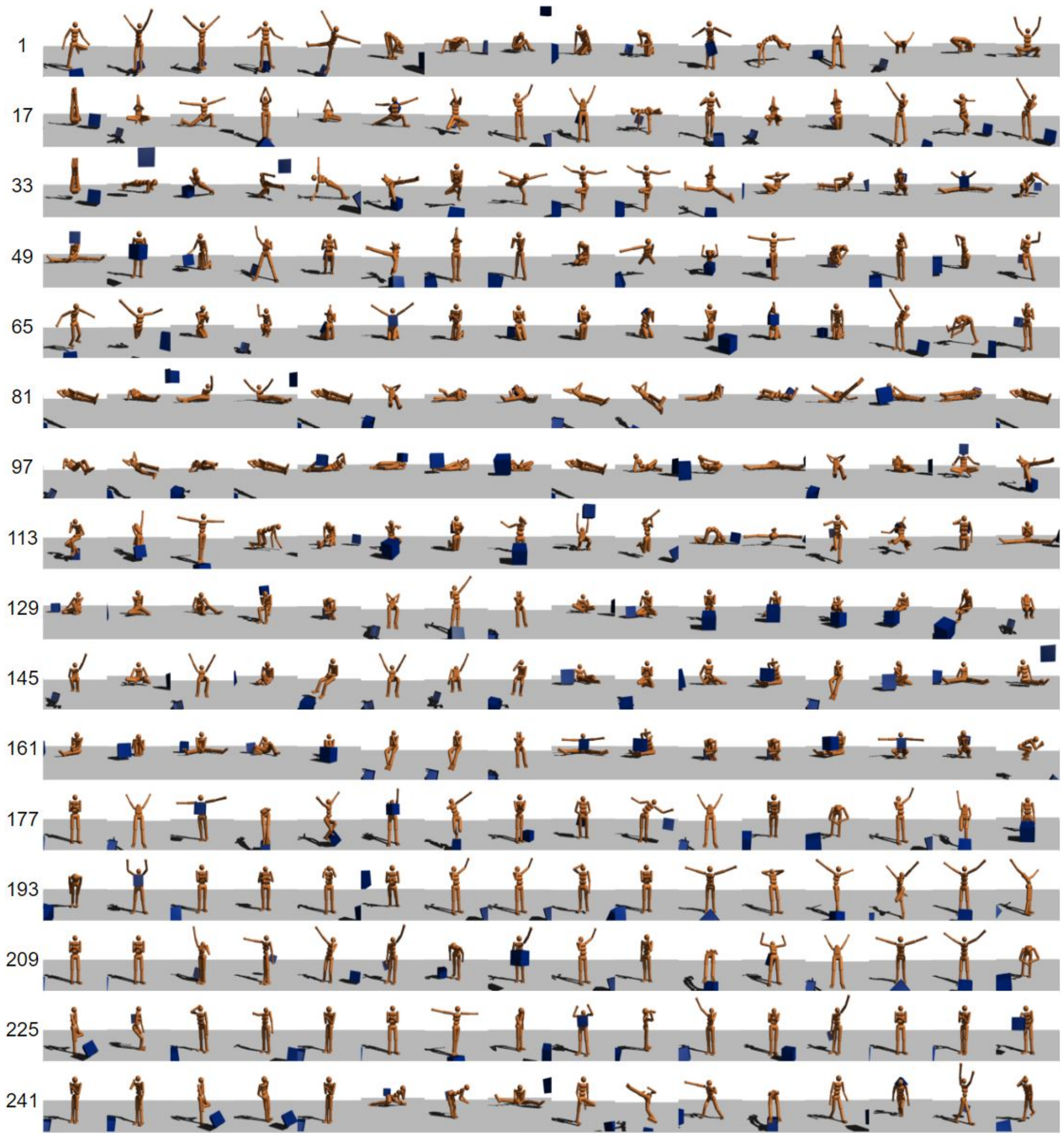


Figure 26. Front views of the highest-scoring configurations retrieved from the single-view embedding-diversity dataset with EVA02-E-14+, using single-view evaluations.

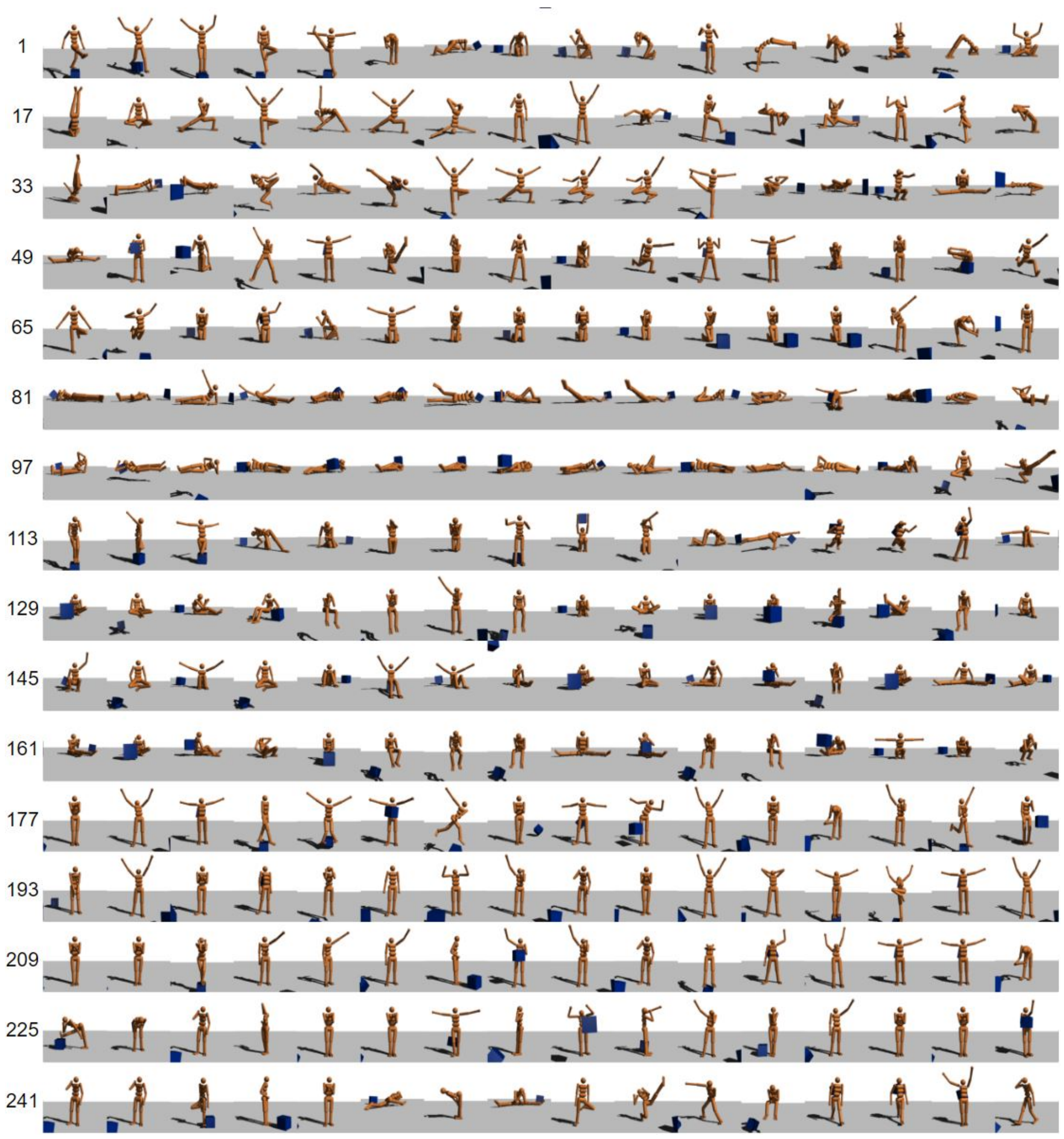


Figure 27. Front views of the highest-scoring configurations retrieved from the single-view embedding-diversity dataset with EVA02-E-14+, using multiview evaluations.

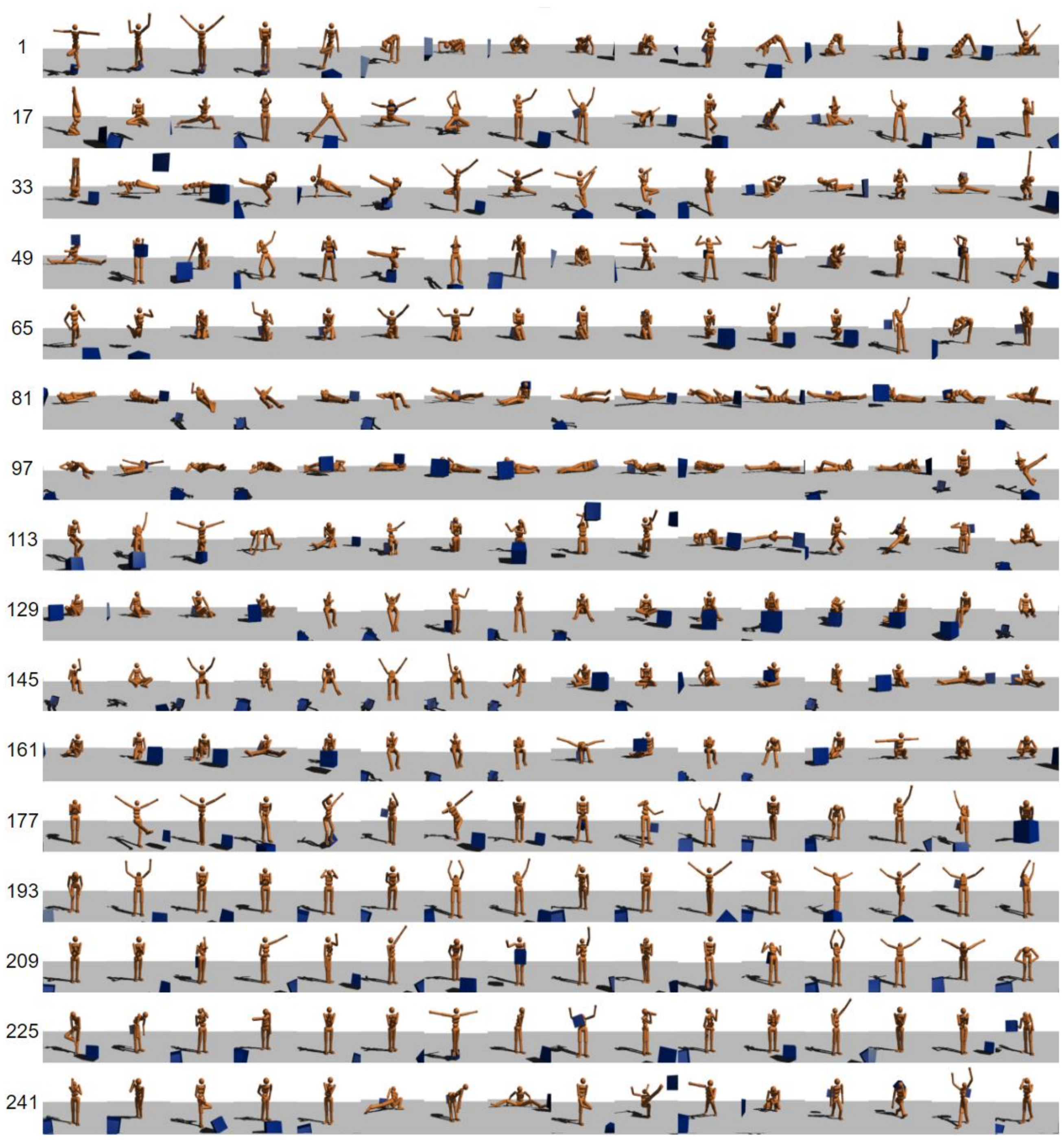


Figure 28. Front views of configurations retrieved from the single-view embedding-diversity dataset, then finetuned using a single-view distilled model.

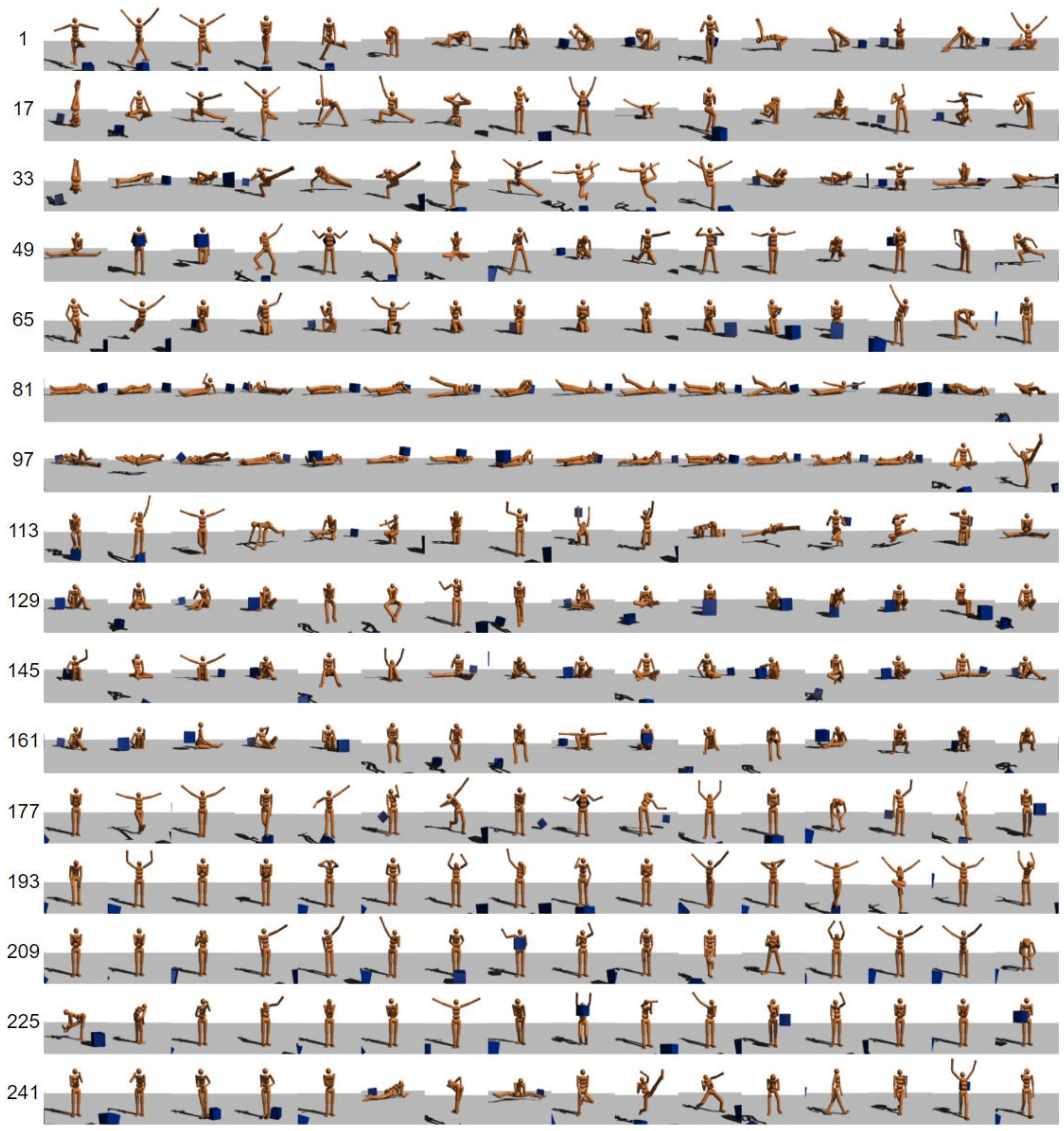


Figure 29. Front views of configurations retrieved from the multiview embedding-diversity dataset, then finetuned using a multiview distilled model.

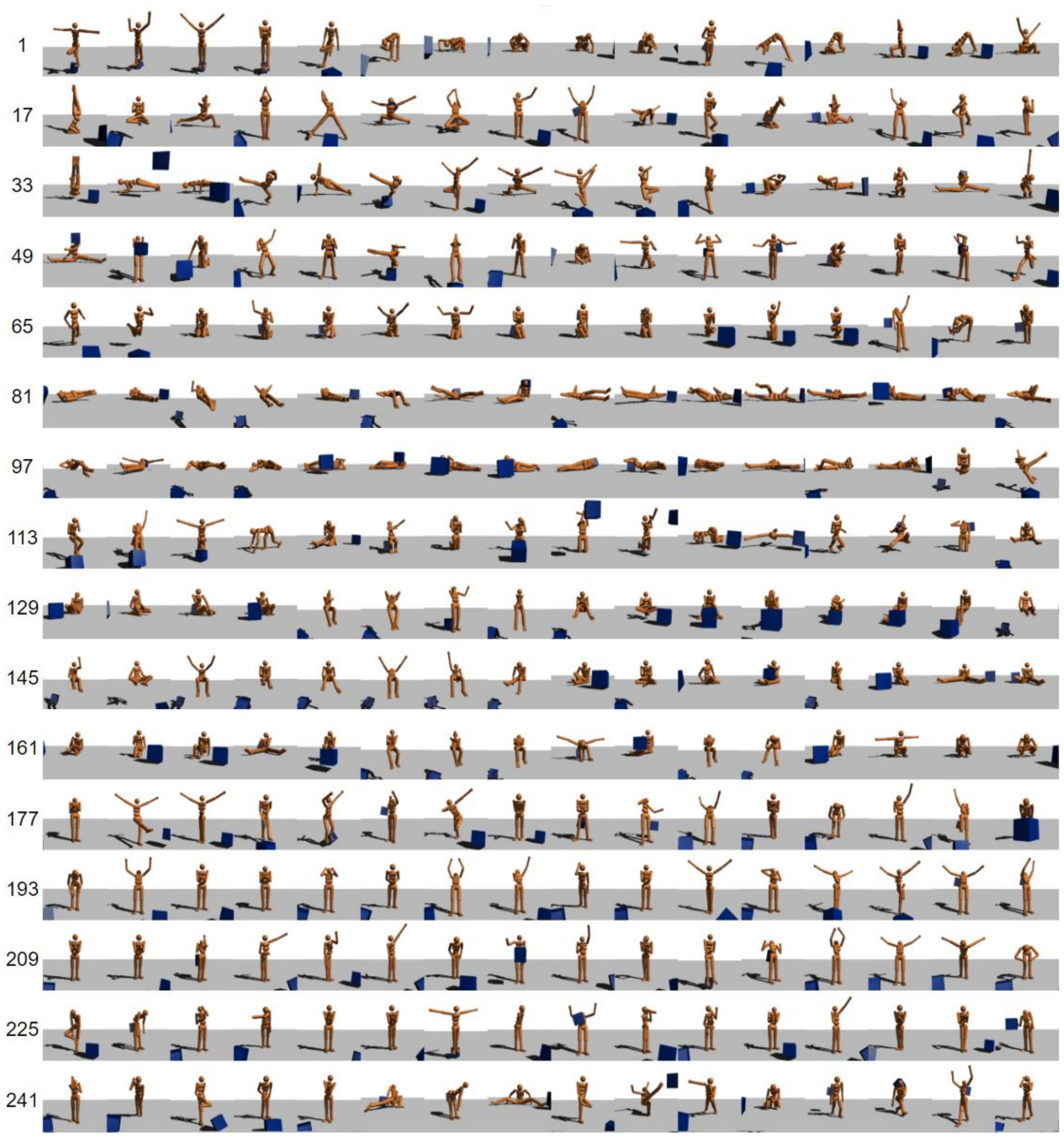


Figure 30. Front views of configurations retrieved from the single-view embedding-diversity dataset, finetuned using a single-view distilled model, then selected using EVA02-E-14+ to evaluate single-view scores.

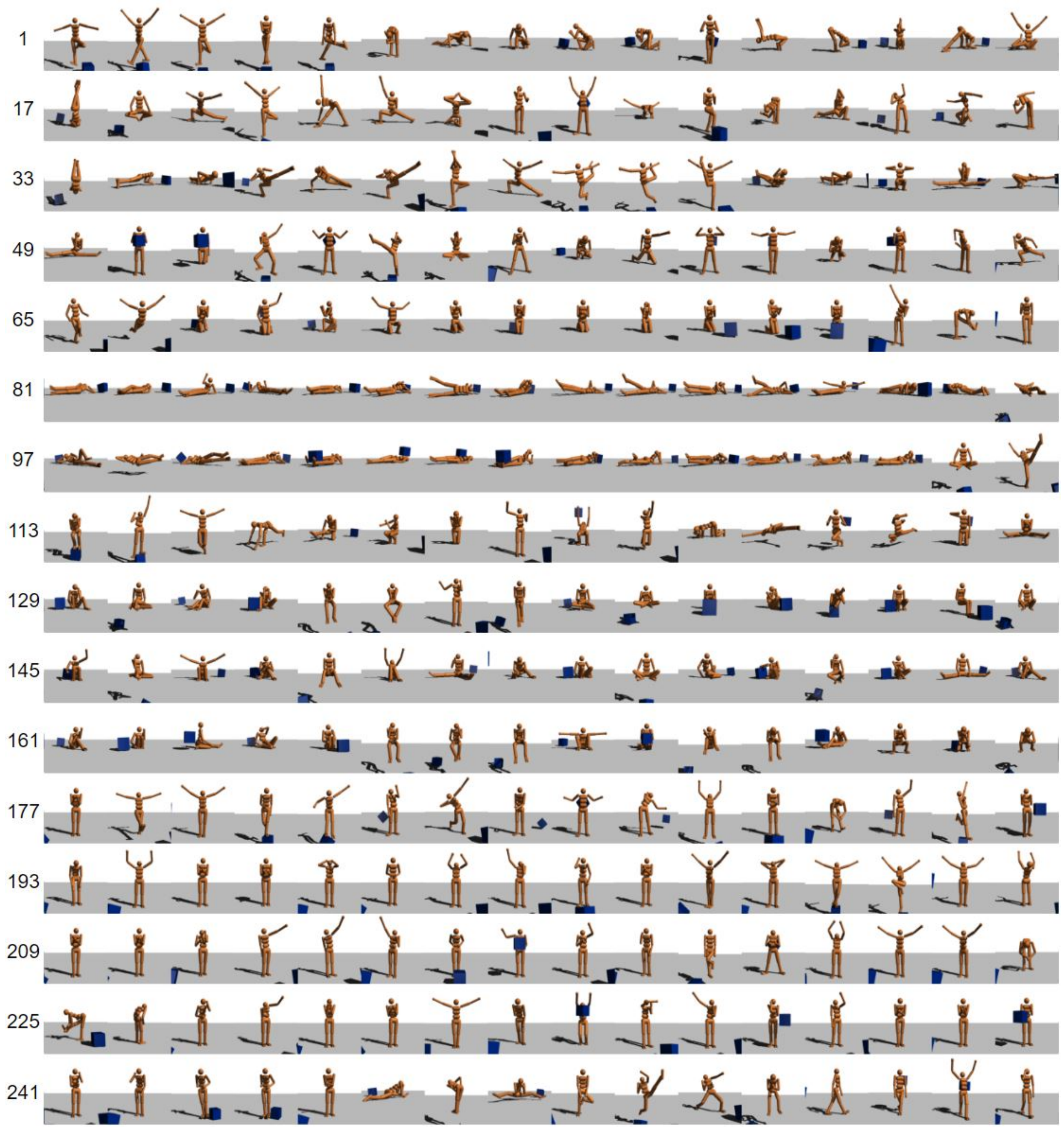


Figure 31. Front views of configurations retrieved from the multiview embedding-diversity dataset, finetuned using a multiview distilled model, then selected using EVA02-E-14+ to evaluate multiview scores.

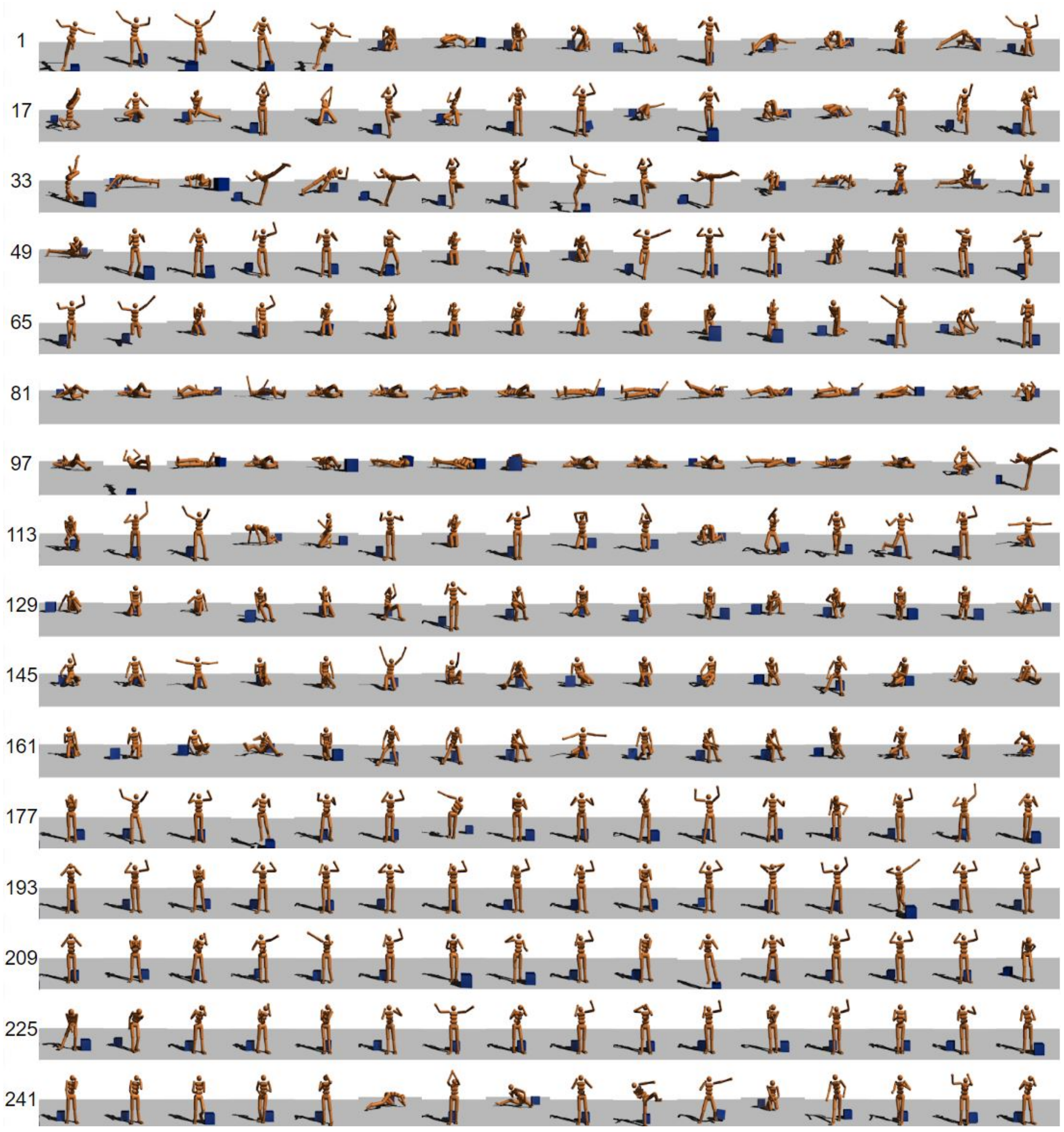


Figure 32. Front views of the best-in-trajectory configurations reached by GCRL-R based on multiview text-configuration score evaluated by EVA02-E-14+.

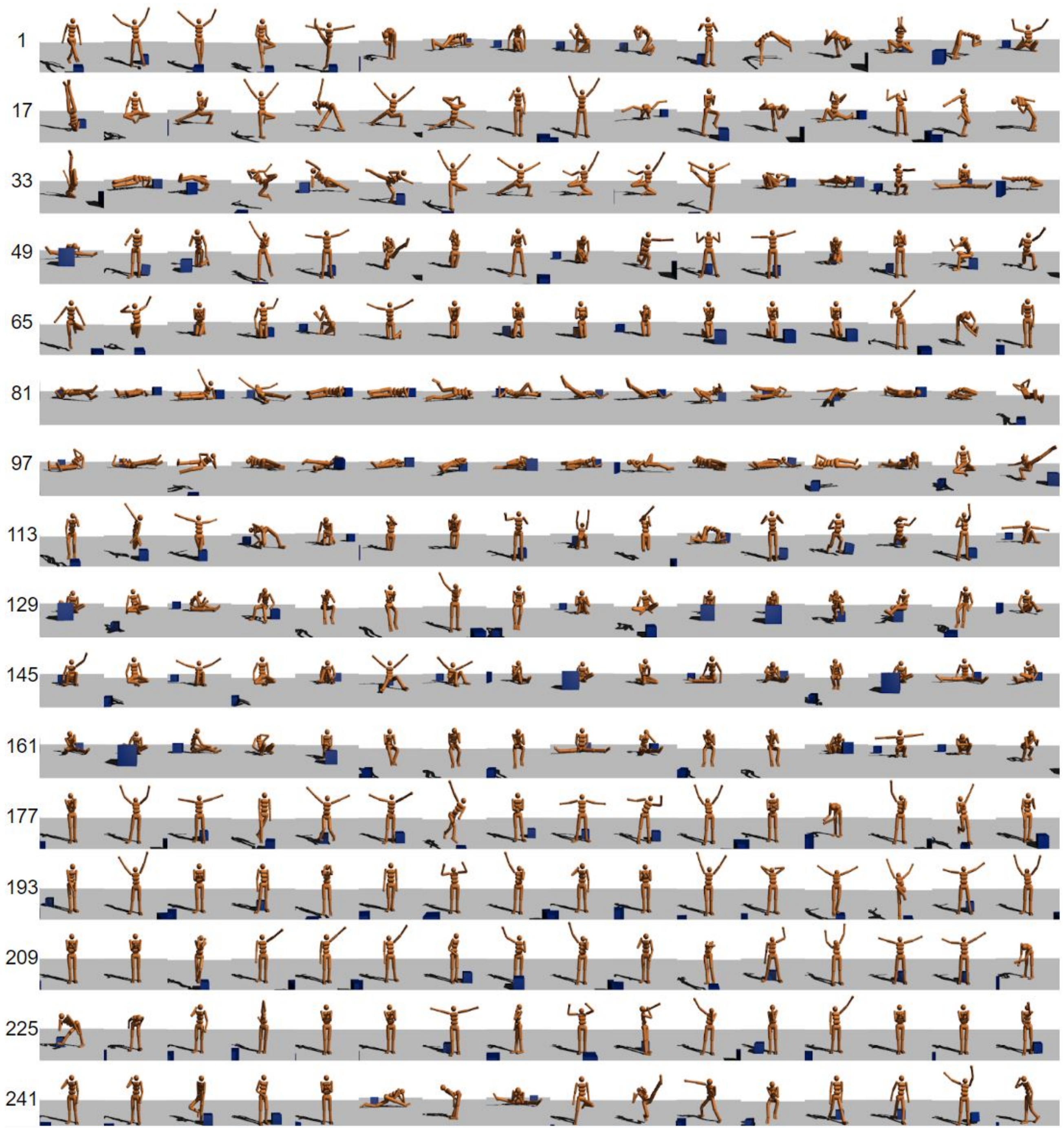


Figure 33. Front views of the best-in-trajectory configurations reached by GCRL-D based on multiview text-configuration score evaluated by EVA02-E-14+.

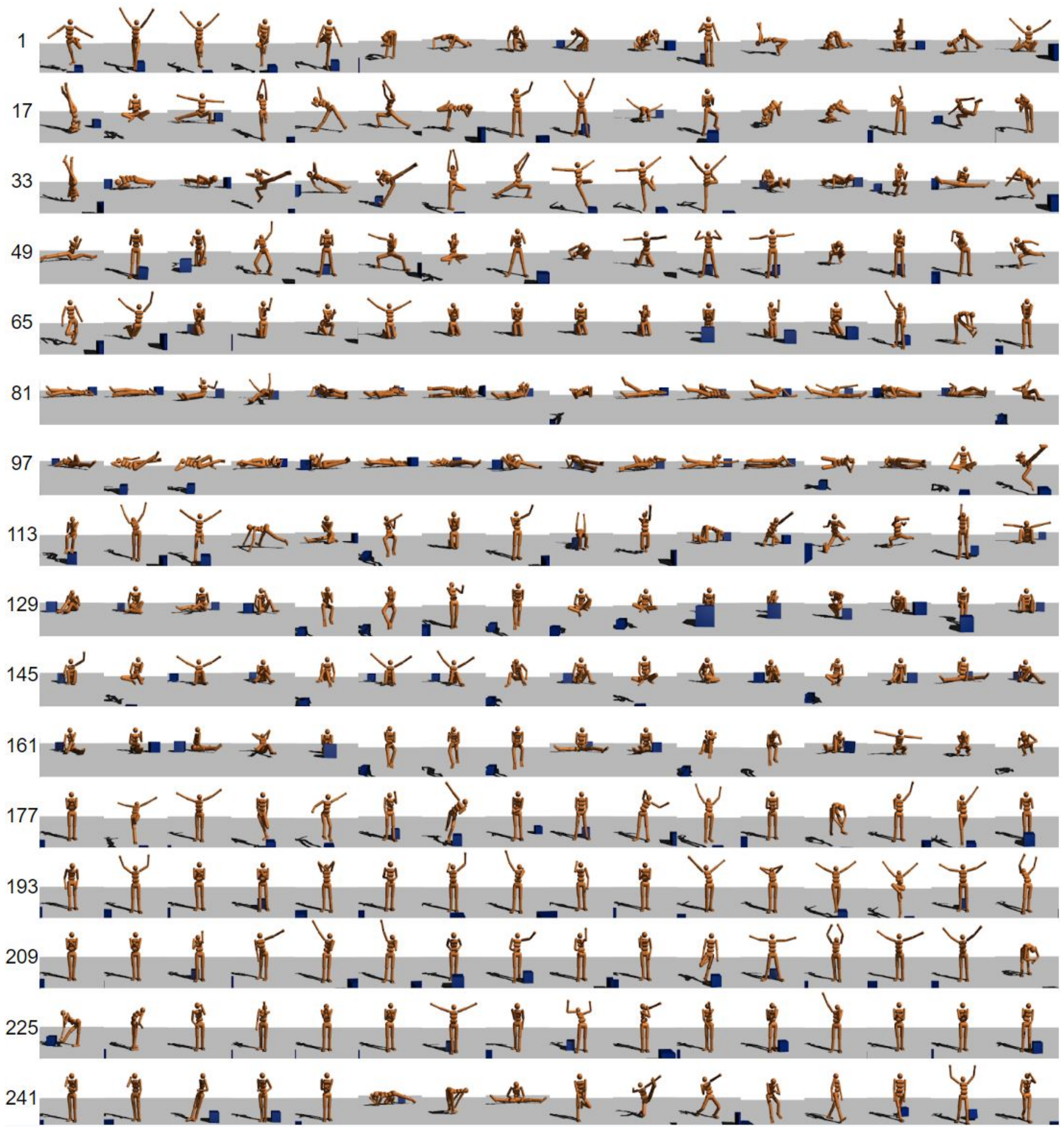


Figure 34. Front views of the best-in-trajectory configurations reached by GCRL-F based on multiview text-configuration score evaluated by EVA02-E-14+.

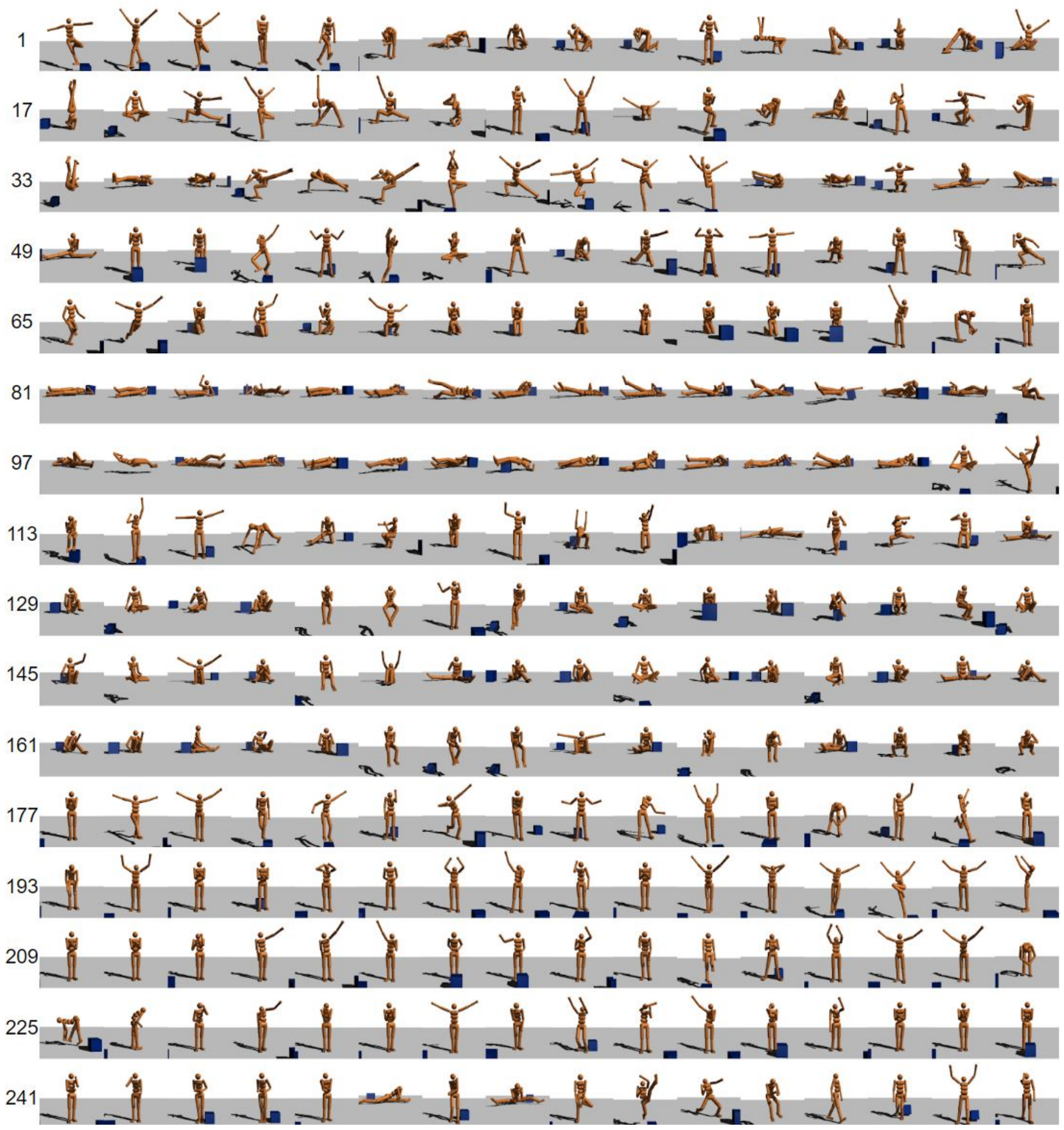


Figure 35. Front views of the best-in-trajectory configurations reached by GCRL-S based on multiview text-configuration score evaluated by EVA02-E-14+.

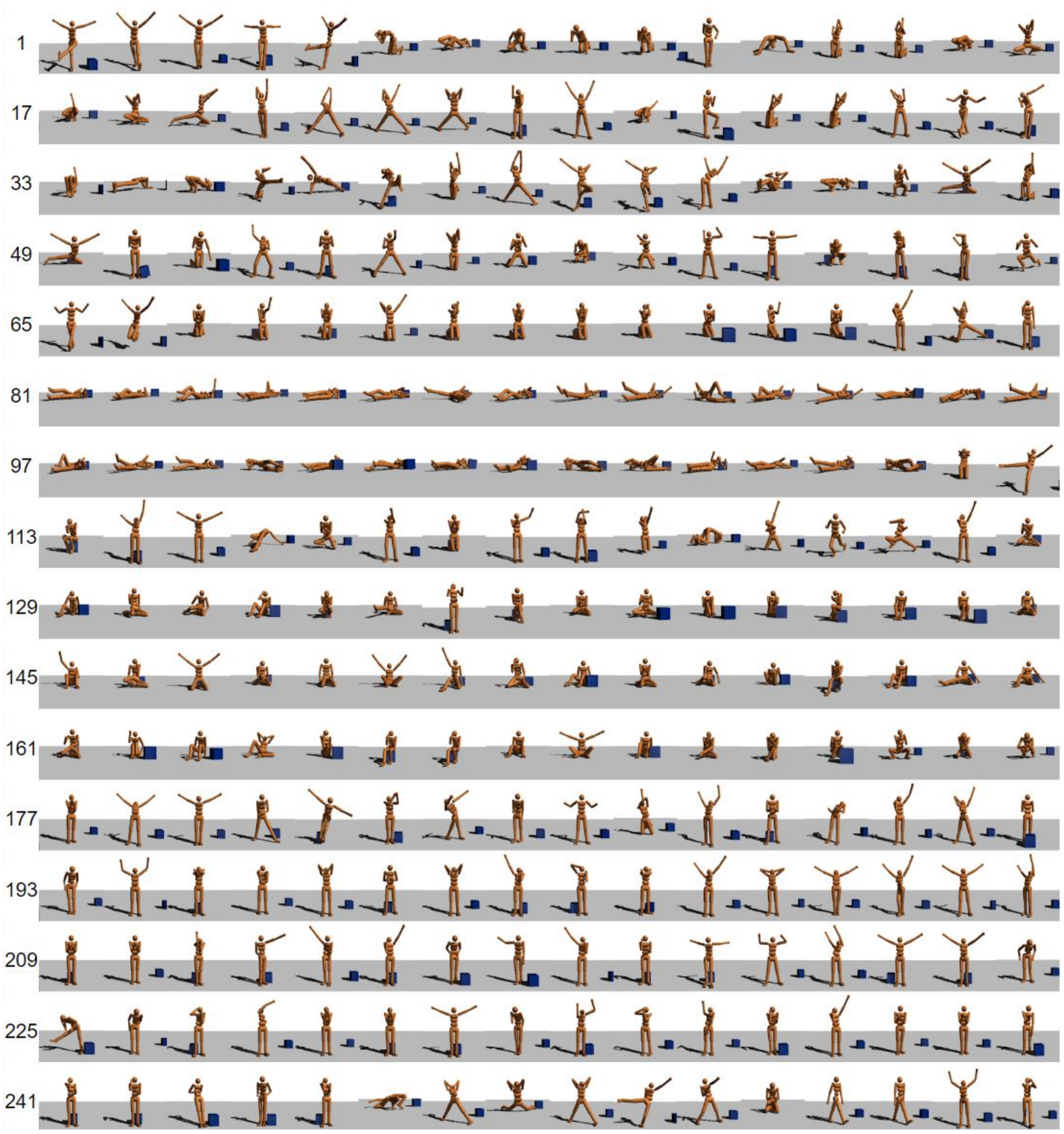


Figure 36. Front views of the best-in-trajectory configurations reached by the MTRL baseline on its training tasks, based on multiview text-configuration score evaluated by EVA02-E-14+.

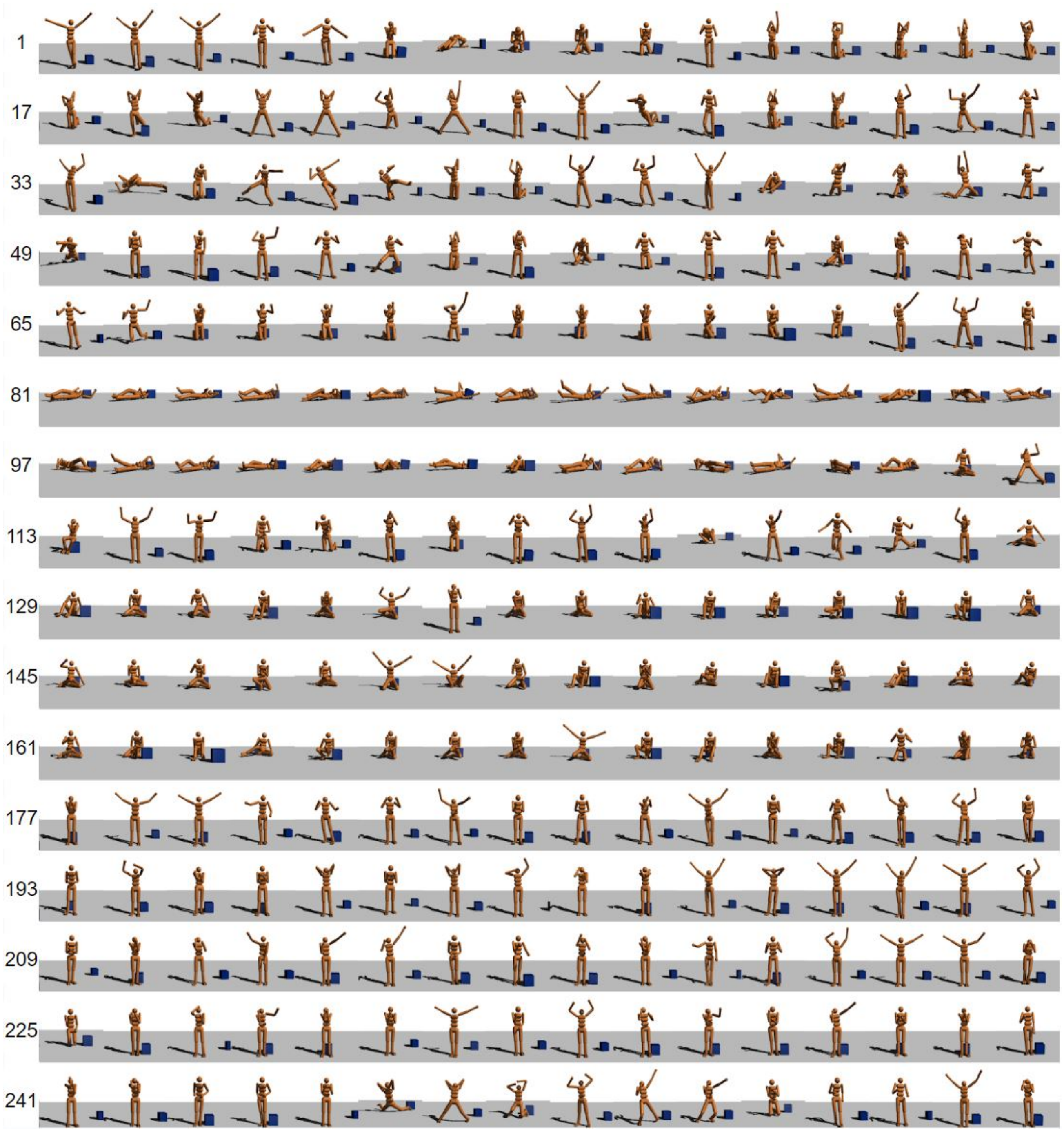


Figure 37. Front views of the best-in-trajectory configurations reached by the MTRL baseline on its test tasks, based on multiview text-configuration score evaluated by EVA02-E-14+.