

Appendix

A Proof For Theorem 1

Theorem 1. Define $J(\pi, r)$ to be the expected return of a policy π under reward r . For an offline dataset d^O with density ρ , a learned log distribution ratio: $\nu(s) = \log(\frac{\rho^E(s)}{\rho(s)})$, $D_{KL}(\rho^\pi, \rho^E) \leq -J(\pi, r^{imit}) + D_{KL}(\rho^\pi(s, a), \rho(s, a))$ where $r^{imit}(s) = \nu(s) \forall s$. The corresponding z_{imit} minimizing the upper bound is given by $z_{imit} = \mathbb{E}_\rho[r^{imit}(s)\varphi(s)] = \mathbb{E}_{\rho^E}[\frac{\nu(s)}{e^{\nu(s)}}\varphi(s)]$ where φ denotes state features learned by the BFM.

Proof. Let ρ be the density of the offline dataset, ρ^π be the visitation distribution w.r.t. policy π and ρ^E be the expert density. The distribution matching objective mentioned in Equation 4 using KL divergence is given as:

$$\min_{\rho^\pi} D_{KL}(\rho^\pi || \rho^E) \quad (5)$$

With simple algebraic manipulation, the divergence can be simplified to,

$$D_{KL}(\rho^\pi || \rho^E) = \mathbb{E}_{\rho^\pi} \left[\log \frac{\rho}{\rho^E} \right] + \mathbb{E}_{\rho^\pi} \left[\log \frac{\rho^\pi}{\rho} \right] \quad (6)$$

$$= \mathbb{E}_{\rho^\pi} \left[\log \frac{\rho(s)}{\rho^E(s)} \right] + D_{KL}(\rho^\pi(s) || \rho(s)) \quad (7)$$

$$= -J(\pi, \log \frac{\rho^E}{\rho}) + D_{KL}(\rho^\pi(s) || \rho(s)) \quad (8)$$

$$\leq -J(\pi, \log \frac{\rho^E}{\rho}) + D_{KL}(\rho^\pi(s, a) || \rho(s, a)) \quad (9)$$

The last line follows from the fact that $D_{KL}(\rho^\pi(s) || \rho(s)) \leq D_{KL}(\rho^\pi(s, a) || \rho(s, a))$.

$$D_{KL}(\rho^\pi(s, a) || \rho(s, a)) = \mathbb{E}_{\rho^\pi(s, a)} \left[\log \frac{\rho^\pi(s, a)}{\rho(s, a)} \right] \quad (10)$$

$$= \mathbb{E}_{\rho^\pi(s, a)} \left[\log \frac{\rho^\pi(s) \pi(a|s)}{\rho(s) \pi^D(a|s)} \right] \quad (11)$$

$$= \mathbb{E}_{\rho^\pi(s, a)} \left[\log \frac{\rho^\pi(s)}{\rho(s)} \right] + \mathbb{E}_{\rho^\pi(s, a)} \left[\log \frac{\pi(a|s)}{\pi^D(a|s)} \right] \quad (12)$$

$$= \mathbb{E}_{\rho^\pi(s)} \left[\log \frac{\rho^\pi(s)}{\rho(s)} \right] + \mathbb{E}_{\rho^\pi(s, a)} \left[\log \frac{\pi(a|s)}{\pi^D(a|s)} \right] \quad (13)$$

$$= D_{KL}(\rho^\pi(s) || \rho(s)) + \mathbb{E}_{s \sim \rho^\pi} [D_{KL}(\pi(a|s) || \pi^D(a|s))] \quad (14)$$

$$\geq D_{KL}(\rho^\pi(s) || \rho(s)) \quad (15)$$

Rewriting the minimization of the upper bound of KL as a maximization problem by reversing signs, we get:

$$\max_{\pi} \left[J(\pi, \log \frac{\rho^E}{\rho}) - D_{KL}(\rho^\pi(s, a) || \rho(s, a)) \right] \quad (16)$$

The first term is an RL objective with a reward function given by $\log(\frac{\rho^E}{\rho})$, and the second term is an offline regularization to constrain the behaviors of offline datasets. Following prior works [35, 46], since our BFM is trained on an offline dataset and limited to output skills in support of dataset actions, and we can ignore the regularization to infer the latent z parameterizing the skill. A heuristic yet performant alternative is to use a shaped reward function of $\frac{\rho^E}{\rho}$, which allows us to avoid training the discriminator completely and was shown to lead to performant imitation in [57]. \square

B Experimental Details

B.1 Environments

B.1.1 DM-control environments

We use continuous control environments from the DeepMind Control Suite [72].

Walker: The agent has a 24 dimensional state space consisting of joint positions and velocities and 6 dimensional action space where each dimension of action lies in $[-1, 1]$. The system represents a planar walker.

Cheetah: The agent has a 17 dimensional state space consisting of joint positions and velocities and 6 dimensional action space where each dimension of action lies in $[-1, 1]$. The system represents a planar biped “cheetah”.

Quadruped: The agent has a 78 dimensional state space consisting of joint positions and velocities and 12 dimensional action space where each dimension of action lies in $[-1, 1]$. The system represents a 3-dimensional ant with 4 legs.

Stickman: Stickman was recently introduced as a task that bears resemblance to a humanoid in [49]. It has a 44 dimensional observation space and a 10 dimensional action space where each dimension of action lies in $[-1, 1]$.

SMPL 3D Humanoid: The agent has a 358 dimensional state space consisting of joint positions and velocities and 69 dimensional action space where each dimension of action lies in $[-1, 1]$. The system represents a 3-dimensional humanoid.

For all the environments we consider image observations of size 64 x 64. All DM Control tasks have an episode length of 1000.

B.2 Evaluation Protocol

To evaluate models for behavior generation through language prompts, we considered a set of 4 prompts per environment. One key consideration in designing these prompts was the generative video model’s capability of generating reasonable imagined trajectories. Due to computing limitations, we were restricted to using a fairly small video embedding (1 billion parameters) and generation model (43 million parameters). The interpretability of our framework allows us to declare failures before they happen by looking at the generations for imagined trajectories.

For the set of task prompts specified by language, there is no ground truth reward function and there does not exist a reliable quantitative metric to verify which of the methods perform better. Instead, since humans communicate their intents via language, humans are the best judge of whether the agent has demonstrated the behavior they intended to convey. In this work we use a Multimodal LLM as a judge, following studies by prior works demonstrating the correlation of LLMs judgment to humans [14]. We use GPT-4o model as the judge, where the GPT-4o model is provided with two videos, one generated by a base method, and another generated by one of the methods we consider, and asked for preference between which video is better explained by the text prompt for the task. When inputting the videos to the judge, we randomize the order of the baseline and proposed methods to reduce the effect of anchoring bias. The prompt we use to compare the two methods is given here:

For prompt to policies:

```
1 response = client.chat.completions.create(  
2     model=MODEL,  
3     messages=[  
4         {"role": "system", "content": "For the given summarization:\n  
5         '{task prompt}', which video is more aligned with the summarization?"},  
6         {"role": "user", "content": [  
7             "Video A",  
8             *map(lambda x: {"type": "image_url",  
9                 "image_url": {"url": f'data:image/jpeg;base64,{x}'\n  
10                 }, video1),  
11             "Video B",  
12             *map(lambda x: {"type": "image_url",
```

```

13         "image_url": {"url": f'data:image/jpg;base64,{x}'\
14         }, video2),
15     "FIRST provide a one-sentence comparison of the two videos\
16     and explain which you feel the given summarization explains better.\
17     SECOND, on a new line, state only 'A' or
18     'B' to indicate\
19     which video is better explained by the given \
20     summarization. Your response should use
21     the format:\
22     Comparison: <one-sentence comparison and explanation>\
23     Better explained by summarization: <'A' or 'B'>"
24     ]
25     }
26
27 ],
28 )

```

For cross-embodiment video to policies:

```

1 cross_embodied_video_description = [*map(lambda x: {"type": "image_url",
2     "image_url": {"url": f'data:image/jpg;base64,{x}'},
3     cross_embodied_video)}]
4
5 response = client.chat.completions.create(
6     model=MODEL,
7     messages=[
8         {"role": "system", "content": f"For the original video:
9         '{cross_embodied_video_description}', which of the
10        following given videos describe a behavior more similar
11        to the original video?"},
12         {"role": "user", "content": [
13             "Video A",
14             *map(lambda x: {"type": "image_url",
15                 "image_url": {"URL":
16                     f'data:image/jpg;base64,{x}',
17                     }, video1),
18             "Video B",
19             *map(lambda x: {"type": "image_url",
20                 "image_url": {"URL":
21                     f'data:image/jpg;base64,{x}',
22                     }, video2),
23             "FIRST provide a one-sentence comparison of
24             the two videos and explain \
25             which you feel matches the behavior
26             shown in original video better .
27             SECOND, on a new line, state only 'A' or \
28             'B' to indicate which video is better aligned
29             to the task demonstrated in the original video.
30             Your response should use \
31             the format:\
32             Comparison: <one-sentence comparison and explanation>\
33             Better matches the original video: <'A' or 'B'>"
34             ]
35             }
36
37     ],
38 )

```

B.3 Dataset Collection for Zero-Shot RL

For Cheetah, Walker, Quadruped, and Stickman environments, our data is collected following a pure exploration algorithm with no extrinsic rewards. In this work, we use intrinsic rewards obtained from Random Network Distillation [10] to collect our dataset based on the protocol by ExoRL [86] and using the implementation from repository [ExoRL repository](#). For Cheetah, Walker,

and Quadraped, our dataset comprises 5000 episodes and equivalently 5 million transitions, and for Stickman, our dataset comprises 10000 episodes or equivalently 10 million transitions. Due to the high dimensionality of action space in Stickman, RND does not discover a lot of meaningful behaviors; hence we additionally augment the dataset with 1000 episodes from the replay buffer of training for a ‘running’ reward function and 1000 episodes of replay buffer trained on a ‘standing’ reward function.

B.4 Baselines

Zero-shot text to policy behavior has not been widely explored in RL literature. However, Offline RL using language-based rewards utilizes an offline dataset to learn policies and is thus zero-shot in terms of rolling out the learned policy. This makes it a meaningful baseline to compare against. Offline RL uses the same MDP formulation as described in Section 3 to learn a policy $\pi : \mathcal{S} \rightarrow \Delta(\mathcal{A})$, given a reward function $r : \mathcal{S} \rightarrow \mathbb{R}$ and offline dataset \mathcal{D} . The offline dataset consists of state, action, next-state, reward transitions $(s, a, s', r(s))$. One of the core challenges of Offline RL is to learn a Q-function that does not overestimate the reward of unseen actions, which then at evaluation causes the agent to drift from the support of the offline dataset \mathcal{D} .

We implement two offline RL baselines to compare with RLZero– Implicit Q-learning (IQL, [37]) and Offline TD3 (TD3, [22]). Both of these methods share the same offline dataset as used to learn the successor measure in RLZero, which is described in Section 5, and gathered using RND. Since these datasets are reward-free, we must still construct a reward function that provides meaningful rewards for an agent achieving the behavior that aligns with the text prompt. Formally, given language instruction $e^l \in \mathcal{E}^l$, frame stack $(o_{t-k}, o_{t-k+1}, \dots, o_t) \in \mathcal{I}$, and embedding VLM $\phi : \mathcal{E} \rightarrow \mathcal{Z}$, which can also embed frame stacks $\phi : \mathcal{I} \rightarrow \mathcal{Z}$ (and where observations $o_i \in \mathcal{I}$, and we use o_i for this section), the reward for a corresponding language instruction and frame stack k is the cosine similarity between the stacked language embedding and the frame embedding:

$$r(o_{t-k:t}, e^l) = \frac{\phi(e^l) \cdot \phi(o_{t-k:t})}{\|\phi(e^l)\| \|\phi(o_{t-k:t})\|} \quad (17)$$

For any individual task, e^l is fixed and this is a reward function dependent on observations (as represented by a frame stack $o_{t-k:t}$). Notice that this representation closely matches that in Equation 3, but instead of finding the optimal sequence of observations, we simply compute reward as the cosine similarity between language and frames. Since the strength of the embedding space is vital to the quality of the reward function for offline RL, we evaluate two different vision-language models:

Image-language reward (SigLIP [89]): take a stack of 3 frames encode them using SigLIP, then the reward is computed as the cosine distance of the embeddings and the SigLIP embedding of language.

Video-language reward (InternVideo2 [79]): this method takes in previous frames $o_{0:t-1}$ as context and uses it to generate an embedding of the current frame observation o_t . The video encoder then takes the cosine similarity of $\phi(o_{0:t})$ and $\phi(e^l)$. This allows the reward function to provide rewards based not only on reaching certain states, but the agent exhibiting temporally extended behaviors that match the behavior. In practice, providing rewards using an image-based encoder for frame stacks can be challenging for tasks such as walking because they require context, and video-based rewards offer a way to better encode the temporal context.

B.4.1 Offline RL

Implicit Q-learning [37] Implicit Q-learning builds on the classic TD error (revised in our context of language-instruction rewards):

$$L(\theta) = E_{(s,a,s',a') \sim \mathcal{D}}[(r(s, e^l) + \gamma Q_{\hat{\theta}}(s', a') - Q_{\theta}(s, a))^2]$$

to learn a Q function Q_{θ} . IQL builds on this loss to handle the challenge of ensuring that the Q-values do not speculate on out-of-distribution actions while also ensuring that the policy is able to exceed the performance of the behavior policy. Exceeding the behavior policy is important because the dataset is collected using RND, meaning that any particular trajectory from the dataset is unlikely to perform well on a language reward. The balance of performance is achieved by optimizing the objective with expectile regression:

$$L_2^{\tau}(u) = |\tau - \mathbb{1}(u < 0)|u^2$$

Where $\tau > 0.5$ is the selected expectile. Expectile regression gives greater weight to the upper expectiles of a distribution, which means that the Q function will focus more on the upper values of the Q function.

Rather than optimize the objective with $Q(s', a')$ directly, IQL uses a value function to reduce variance to give the following objectives:

$$L_V(\psi) = E_{(s,a) \sim \mathcal{D}}[L_2^\tau(Q_\theta(s, a)V_\psi(s))]$$

$$L_Q(\theta) = E_{(s,a,s',a') \sim \mathcal{D}}[L_2^\tau(r(s, e^l) + V_\psi(s') - Q_\theta(s, a))]$$

Using the Q-function, a policy can be extracted using advantage weighted regression:

$$L(\phi) = E_{(s,a) \sim \mathcal{D}}[\exp(\beta(Q_\theta(s, a) - V_\psi(s))) \log \pi_\phi(a|s)].$$

Where β is the inverse temperature for the advantage term.

TD3 [23]:

TD3 was demonstrated to be the best performing algorithm when learning from exploratory RND datasets in [86]. While TD3 does not explicitly address the challenges discussed in implicit Q-learning and learns using Bellman Optimality backups, the approach is simple and works well in practice. The algorithm uses a deterministic policy extraction $\pi : \mathcal{S} \rightarrow \mathcal{A}$ to give the following objective:

$$\pi = \arg \max_{\pi} E_{(s,a) \sim \mathcal{D}}[Q(s, \pi(s))]$$

B.5 RLZero

Algorithm 1 RLZero

- 1: Init: Pretrained Video Generation Model VM , Pretrained BFM π_z , Offline Exploration Dataset d^O
 - 2: Given: text prompt t
 - 3: Generate imagination video given the text prompt: $\{i_1, i_2, \dots, i_l\} = VM(t)$
 - 4: Project the imagined frames to real observations using embedding similarity as in Eq 3.
 - 5: Use Theorem 1 for zero-shot inference to obtain $BFM(\{s_1, s_2, \dots, s_l\}) = z_{imit}$ and return $\pi_{z_{imit}}$.
-

B.5.1 Text to Imagined Behavior with Video models

To generate a proposed video frame sequence, we utilize the GenRL architecture and provide the workflow using equations from the original paper [49]. First, the desired text prompt is embedded with the underlying video foundation model InternVideo2 [79] $e^{(l)} = f_{PT}^{(l)}(y)$. These embeddings are then repeated n_{frames} times (we use $n_{frames} = 32$) to match the temporal structure expected by the world model. The repeated text embeddings are passed through an aligner module $e(v) = f_\psi(e^{(l)})$. The aligner is implemented as a UNet and it is used to address the multimodality gap [40] when embeddings from different modalities occupy distinct regions in the latent space. Next, the aligned video embeddings are concatenated with temporal embeddings. The temporal embeddings are one-hot encodings of the time step modulo n_{frames} providing frame-level positional information. The first embedding is passed to the world model connector $p_\psi(s_t|e)$ to initialize the latent state. For each subsequent time step, the sequence model $h_t = f_\phi(s_{t-1}, a_{t-1}, h_{t-1})$ (implemented as a GRU) updates the deterministic state h_t . The deterministic state h_t is mapped to a stochastic latent state (s_t) using the dynamics predictor $p_\phi(s_t|h_t)$. The dynamics predictor, implemented as an ensemble of MLPs, predicts the sufficient statistics (mean and standard deviation) for a Normal distribution over s_t . During inference, the mean of this distribution is used as the latent state. Finally, the latent state s_t is passed to a convolutional decoder $p_\phi(x_t|s_t)$ to reconstruct the video frame x_t . This process is repeated for all time steps ($t = 1, \dots, n_{frames}$).

B.5.2 Grounding imagined observations to observations in offline dataset

As described in Section 4, we ground imagined sequences by retrieving real offline states based on similarity in an embedding space. This enables us to create a suitable z -vector for distribution matching which is the expected value of the state features under the distribution of imagined states ($\rho_{imagined}$). During our dataset collection phase, we save both the agent’s proprioceptive state as well as the corresponding rendered images and search over the images to then find the corresponding

state. Our code supports both stacked-frame embeddings and single-frame embeddings. We find that stacked-frame embeddings were helpful in modeling temporal dependencies through velocity and acceleration, which are crucial for recreating the intended behavior. SigLIP [89], which replaces CLIP’s [60] softmax-based contrastive loss with a pairwise sigmoid loss, resulted in qualitatively better matches to exact positions within sequences, imitating behavior more accurately than CLIP. For both models, we use the OpenCLIP [30] framework. Our matching process first involves precomputing embeddings offline, which are stored in chunks of up to 100,000 frames to optimize memory usage and retrieval speed. During inference, we load this file and embed the query frame sequence from GenRL [49] into the same latent space. We process these query embeddings by dividing them into chunks of k -frame sequences (k is generally 3 or 5), where each sequence consists of the current frame and the $k - 1$ preceding frames. If there are not enough preceding frames, we repeat the first frame to fill the gap. For each chunk of saved embeddings, we compute dot products between the query chunk and all subsequences of size k in the saved embeddings. We track the highest similarity score for each query chunk and return the frames corresponding to the closest embedding sequences.

B.5.3 Training a zero-shot RL agent

In this work, we chose Forward-Backward (FB) [76] as our zero-shot RL algorithm and trained it on proprioceptive inputs. Our implementation follows closely from the author’s codebase. Specifically, FB trains Forward, Backward, and Actor networks. The backward networks are used to map a demonstration or a reward function to a skill, which is then used to learn a latent-conditional Actor. Our experiments were performed on NVIDIA-A40 and AMD EPYC 7763 64-Core Processor machine. The hyperparameters for our FB implementation are listed below:

Implementation: We build upon the codebase for FB https://github.com/facebookresearch/controllable_agent and implement all the algorithms under a uniform setup for network architectures and the same hyperparameters for shared modules across the algorithms. We keep the same method-agnostic hyperparameters and use the author-suggested method-specific hyperparameters. The hyperparameters for all methods can be found in Table 3:

Table 3: Hyperparameters for zero-shot RL with FB.

Hyperparameter	Value
Replay buffer size	$5 \times 10^6, 10 \times 10^6$ (for stickman)
Representation dimension	128
Batch size	1024
Discount factor γ	0.98
Optimizer	Adam
Learning rate	3×10^{-4}
Momentum coefficient for target networks	0.99
Stddev σ for policy smoothing	0.2
Truncation level for policy smoothing	0.3
Number of gradient steps	2×10^6
Regularization weight for orthonormality loss (ensures diversity)	1
FB specific hyperparameters	
Hidden units (F)	1024
Number of layers (F)	3
Hidden units (b)	256
Number of layers (b)	2

B.6 Imagination-Free RLZero

In this section, we propose an alternate method (Figure 5) for mapping a task description into a usable policy. Instead of first embedding a text prompt e^ℓ , generating a video, then mapping the video to a policy parametrization, we propose to map the text prompt directly to a policy parametrization. To do this, we learn a latent mapper $m : e \rightarrow z_{\text{imitation}}$ that relates the latent space of a ViLM to the latent space of our policy parametrization. The mapper is a 3 layer MLP with hidden size of 512.

Pretraining: We first generate a dataset of episodes containing diverse behaviors by rolling out the

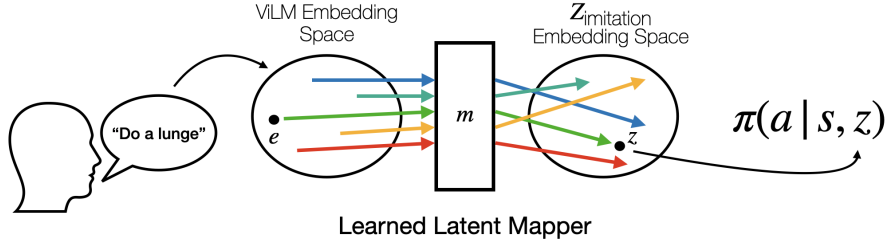


Figure 5: Illustrative diagram of imagination-free RLZero inference

behavior foundation model conditioned on a uniformly random sampled z . The resulting image observation sequences are then down-sampled (by 8) and sliced to break up each episode into smaller chunks of length 8; this preprocessing step helps increase the behavioral diversity and improves the ability of the ViFM to capture semantic meaning. The resulting clips are then embedded using a ViFM (InternVideo2 [79]) where each embedding is denoted by e (as in Section 5.3). Now, we have a set of sequences of length 8 consisting of image observation along with their proprioceptive states, and the embedding for the image sequence.

Now, an obvious option is to map the embedding of image sequence to the z that generated the trajectory. Unfortunately, the way BFM are trained, they do not account for optimal policy invariance to reward functions. That is multiple reward functions that induce the same optimal policy are mapped to different encodings in the Z -space. This presents a problem for the latent mapper, as it becomes a one-to-many mapping for any language encoding. We present an alternative solution which ensures that only one target z is used for a given distribution of states induced by a language encoding. To achieve this we turn back to the imitation learning objective where the sequence of proprioceptive states is used to obtain a policy representation using Lemma 1 which gives the latent z corresponding to the policy that minimizes the distribution divergence to the sequence of given states. We refer to the policy representation embedding space from the Forward-Backward representation as $Z_{\text{imitation}}$ -space.

When optimizing the latent mapper m , we minimize the following loss:

$$\mathcal{L}(\mathcal{D}, m) = \mathbb{E}_{(z_{\text{imitation}}, e) \sim \mathcal{D}} \left[-\frac{m(e) \cdot z_{\text{imitation}}}{\|m(e)\| \cdot \|z_{\text{imitation}}\|} \right]$$

The latent space of the Backward representation is aligned with the latent space of the policy parametrization, so learning a mapping from the ViFM space to the Backward space is equivalent to learning a mapping from the ViFM space to the policy parametrization space.

Inference: During inference, the language prompt is embedded to a latent vector e^l . A known issue with multimodal embedding models is the embedding gap [40], which makes the video embeddings unaligned with text embeddings. To account for this gap, we use an aligner trained in an unsupervised fashion from previous work [49] to align the language embedding (e_{aligned}^l). Then the aligned embedding is passed through the latent mapper to get the policy conditioning $z_{\text{imitation}}$ which gives us the policy that achieves the desired behavior specified through language.

C Additional Results

C.1 Visualizations



Figure 6: **Example Imagined Trajectories:** The video model imagines frames conditioned on the task specified as a text prompt ‘do lunges’.

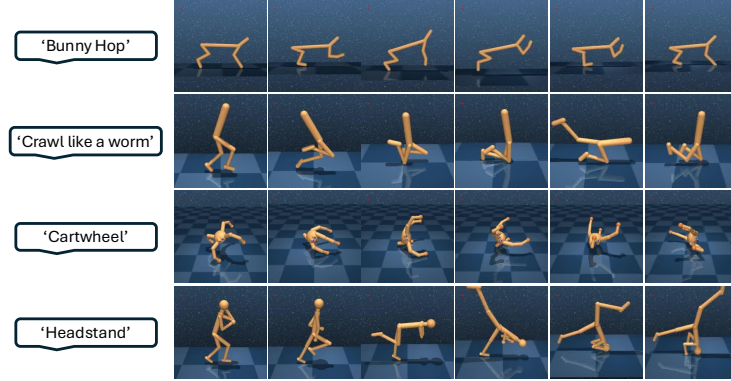


Figure 7: RLZero in action: Qualitative examples of RL converting the given language prompts into behaviors across different domains. Top to bottom: Cheetah, Walker, Quadruped, Stickman.

C.2 Zero-shot imitation: Discriminator vs Discriminator-free

We experiment whether optimizing a tighter bound to KL divergence at the expense of training an additional discriminator in Lemma 1 leads to performance improvements. Table 4 shows that using a discriminator does not lead to a performance improvement and a training-free inference time solution achieves a slightly higher win rate.

C.3 Cross Embodiment experiments

Tables 5 and 6 describe the videos used for cross-embodiment along with the win rate of the behaviors generated by RLZero when compared to a base model which trains SMODICE [46] on the nearest states found with the same grounding methods as RLZero.

C.4 Imagination-Free RLZero Complete Results

We consider an ablation of our method by understanding the need for imagination by replacing the step with an end-to-end learning alternative. This is a novel baseline described in Appendix B.6. Table 7 shows the results of this end-to-end alternative which maps the shared latent space of video language models to behavior policy.

C.5 More Failure Cases

We include more failure cases in Figure 8 and Figure 9 as they can help in understanding the limitations of RLZero better and may inform future work.

C.6 RLZero evaluation with Video-Embedding similarity

In this section, we experiment with another metric for comparison – embedding similarity between a video of the generated behavior and the text. We use InternVideo2 to embed the videos and take the cosine similarity with the prompt used to generate the behavior. Table 8 shows the results for this metric of comparison. Unfortunately, we observed that the similarity score is frequently higher even for behaviors that differ significantly from the prompt. This points to a limitation of using this metric for evaluation. Some reasons for this failure could be the limited context length of 8 for the video embedding model or a misalignment between video and text embedding vectors [40].

	RLZero with discriminator	RLZero
Walker		
Lying Down	5/5	5/5
Walk like a human	5/5	5/5
Run like a human	5/5	5/5
Do lunges	5/5	5/5
Cartwheel	5/5	4/5
Strut like a horse	5/5	5/5
Crawl like a worm	1/5	3/5
Quadruped		
Cartwheel	4/5	4/5
Dance	5/5	5/5
Walk using three legs	4/5	5/5
Balancing on two legs	4/5	5/5
Lie still	2/5	2/5
Handstand	4/5	3/5
Cheetah		
Lie down	1/5	2/5
Bunny hop	5/5	5/5
Jump high	5/5	5/5
Jump on back legs and backflip	5/5	5/5
Quadruped walk	2/5	4/5
Stand in place like a dog	4/5	3/5
Stickman		
Lie down stable	5/5	4/5
Lunges	5/5	5/5
Praying	4/5	4/5
Headstand	5/5	4/5
Punch	4/5	4/5
Plank	4/5	3/5
Average	82.4%	83.2%

Table 4: Win rates computed by GPT-4o of policies trained by different methods when compared to base policies trained by TD3+Image-language reward.

	Prompt Descriptions	Video Link/Meta AI Prompt	Win rate vs SMODICE
Stickman (2D Humanoid)	Human in backflip position	animated human trying backflip	2/5
	Downward facing dog yoga pose	right profile of yoga pose downward facing dog	1/5
	Cow yoga pose	[51]	5/5
	Downward dog with one leg raised in the air	[51]	5/5
	Lying on back with one leg raised in the air	[53]	5/5
	Lying on back with both legs raised in the air	[41]	5/5
	High plank yoga pose	[81]	5/5
	Sitting down with legs laid in the front	[11]	5/5
	Warrior III pose	[87]	3/5
	Front splits	[70]	4/5

Table 5: Comparison of Win rates vs SMODICE for Stickman

	Prompt Descriptions	Video Link/Meta AI Prompt	Win rate vs SMODICE
SMPL (3D Humanoid)	A karate kick position	a karate kick	5/5
	A cat doing a handstand	a side profile of cat doing headstand	5/5
	An arabesque ballet position	ballet movement	5/5
	Animated wikiHow demo of a cartwheel	[83]	5/5
	Running	running	5/5
	Lying crunches	[82]	5/5
	Plank position	[81]	5/5

Table 6: Comparison of Win rates vs SMODICE for 3D SMPL Humanoid

Environment/Task	RLZero	RLZero (Imagination-Free)
Walker		
Lying Down	5/5	5/5
Walk like a human	5/5	4/5
Run like a human	5/5	1/5
Do lunges	5/5	5/5
Cartwheel	4/5	5/5
Strut like a horse	5/5	3/5
Crawl like a worm	3/5	0/5

Table 7: Win rates computed by GPT-4o of policies trained by different methods when compared to base policies trained by TD3+Image-language reward. RLZero shows marked improvement over using embedding cosine similarity as reward functions.



Figure 8: More examples of failed imagination by the video generation model used in RLZero. From top to bottom: Walker - ‘kick’, Quadruped - ‘bunny hop’, Cheetah - ‘frontroll’, Stickman - ‘raise hands while standing in place’

	Image-language reward		Video-language reward		RLZero
	IQL	TD3 (Base Model)	TD3	IQL	
Walker					
Lying Down	2/5 (0.95±0.00)	- (0.89±0.02)	2/5 (0.93±0.01)	5/5 (0.94±0.01)	5/5 (0.93±0.00)
Walk like a human	1/5 (0.93±0.00)	- (0.83±0.02)	3/5 (0.92±0.01)	4/5 (0.94±0.00)	5/5 (0.98±0.00)
Run like a human	5/5 (0.95±0.02)	- (0.88±0.03)	1/5 (0.91±0.01)	2/5 (0.94±0.00)	5/5 (0.96±0.00)
Do lunges	4/5 (0.94±0.01)	- (0.91±0.02)	2/5 (0.92±0.00)	3/5 (0.93±0.00)	5/5 (0.94±0.01)
Cartwheel	4/5 (0.95±0.01)	- (0.93±0.01)	3/5 (0.94±0.01)	4/5 (0.96±0.01)	4/5 (0.95±0.01)
Strut like a horse	5/5 (0.96±0.00)	- (0.94±0.02)	1/5 (0.94±0.00)	3/5 (0.96±0.03)	5/5 (0.96±0.00)
Crawl like a worm	4/5 (0.93±0.00)	- (0.92±0.01)	1/5 (0.92±0.01)	2/5 (0.95±0.01)	3/5 (0.89±0.01)
Quadruped					
Cartwheel	1/5 (0.95±0.00)	- (0.95±0.00)	3/5 (0.95±0.01)	1/5 (0.95±0.01)	4/5 (0.92±0.02)
Dance	5/5 (0.94±0.00)	- (0.94±0.00)	3/5 (0.94±0.02)	1/5 (0.94±0.01)	5/5 (0.93±0.01)
Walk using three legs	2/5 (0.92±0.00)	- (0.91±0.00)	2/5 (0.91±0.01)	3/5 (0.93±0.01)	5/5 (0.93±0.01)
Balancing on two legs	2/5 (0.93±0.01)	- (0.93±0.00)	2/5 (0.93±0.01)	2/5 (0.93±0.00)	5/5 (0.94±0.02)
Lie still	1/5 (0.87±0.00)	- (0.90±0.01)	3/5 (0.94±0.00)	2/5 (0.95±0.00)	2/5 (0.92±0.00)
Handstand	2/5 (0.91±0.01)	- (0.91±0.02)	4/5 (0.92±0.01)	2/5 (0.94±0.00)	3/5 (0.91±0.00)
Cheetah					
Lie down	3/5 (0.92±0.02)	- (0.87±0.00)	2/5 (0.94±0.00)	3/5 (0.94±0.01)	2/5 (0.90±0.01)
Bunny hop	3/5 (0.98±0.00)	- (0.98±0.00)	1/5 (0.98±0.00)	3/5 (0.97±0.02)	5/5 (0.96±0.00)
Jump high	3/5 (0.94±0.01)	- (0.94±0.01)	0/5 (0.94±0.01)	5/5 (0.93±0.01)	5/5 (0.93±0.01)
Jump on back legs and backflip	3/5 (0.93±0.01)	- (0.92±0.00)	0/5 (0.91±0.01)	2/5 (0.92±0.01)	5/5 (0.91±0.01)
Quadruped walk	3/5 (0.96±0.02)	- (0.85±0.01)	3/5 (0.98±0.00)	3/5 (0.99±0.01)	4/5 (0.97±0.01)
Stand in place like a dog	4/5 (0.93±0.01)	- (0.88±0.00)	3/5 (0.98±0.01)	0/5 (0.98±0.00)	3/5 (0.97±0.00)
Stickman					
Lie down stable	2/5 (0.92±0.00)	- (0.91±0.01)	4/5 (0.93±0.00)	1/5 (0.93±0.00)	4/5 (0.91±0.00)
Lunges	0/5 (0.92±0.00)	- (0.93±0.02)	2/5 (0.92±0.01)	0/5 (0.92±0.00)	5/5 (0.96±0.00)
Praying	1/5 (0.85±0.00)	- (0.89±0.02)	0/5 (0.87±0.01)	0/5 (0.87±0.01)	4/5 (0.91±0.00)
Headstand	2/5 (0.90±0.01)	- (0.90±0.00)	2/5 (0.90±0.01)	1/5 (0.87±0.01)	4/5 (0.90±0.00)
Punch	2/5 (0.89±0.02)	- (0.88±0.02)	3/5 (0.88±0.02)	4/5 (0.91±0.00)	4/5 (0.90±0.02)
Plank	0/5 (0.90±0.01)	- (0.93±0.03)	0/5 (0.89±0.01)	0/5 (0.93±0.00)	3/5 (0.96±0.00)
Average	51.2% (0.926)	Base Model (0.908)	40% (0.927)	44.8% (0.936)	83.2% (0.933)

Table 8: Win rates computed by GPT-4o of policies trained by different methods when compared to a base policies trained by TD3+Image-language reward. RLZero shows marked improvement over using embedding cosine similarity as reward functions.



Figure 9: More examples of failed grounding by the image retrieval model used in RLZero. The top image shows the imagined frame or frame from the embodied video, and the bottom is the nearest frame obtained from the agent’s prior interaction dataset.