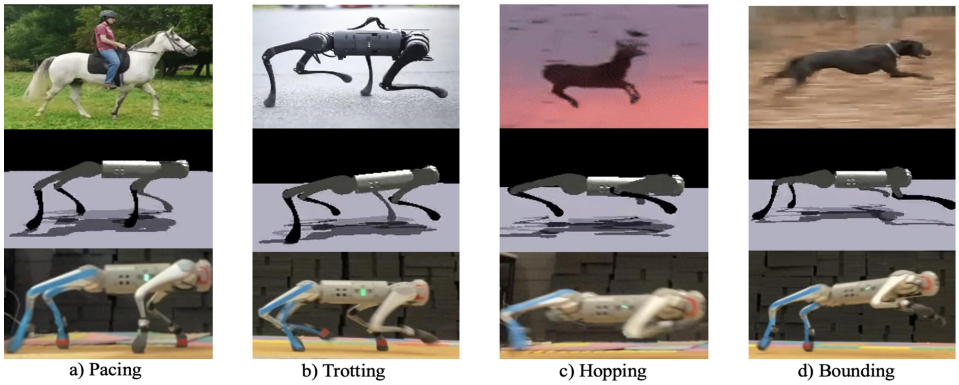
SDS – See it, Do it, Sorted: Quadruped Skill Synthesis from Single Video Demonstration

Maria Stamatopoulou* **Jeffrey Li*** **Dimitrios Kanoulas**
Robot Perception Lab, University College London, United Kingdom
{maria.stamatopoulou.21, jeffrey.li.20, d.kanoulas}@ucl.ac.uk

Abstract: Imagine a robot learning locomotion skills from any single video, without labels or reward engineering. We introduce **SDS** ("See it. Do it. Sorted."), an automated pipeline for skill acquisition from unstructured demonstrations. Using GPT-4o, SDS applies novel prompting techniques, in the form of spatio-temporal grid-based visual encoding (G_v) and structured input decomposition (**SUS**). These produce executable reward functions ($\mathcal{RF}$) from the raw input videos. The $\mathcal{RF}$s are used to train PPO policies and are optimized through closed-loop evolution, using training footage and performance metrics as self-supervised signals. SDS allows quadrupeds (e.g. Unitree Go1) to learn four gaits – *trot*, *bound*, *pace*, and *hop* – achieving 100% gait matching fidelity, Dynamic Time Warping (DTW) distance in the order of 10^{-6}, and stable locomotion with zero failures, both in simulation and the real world. SDS generalizes to morphologically different quadrupeds (e.g. ANYmal) and outperforms prior work in data efficiency, training time and engineering effort. Further materials and the code are open-source under: https://rpl-cs-ucl.github.io/SDSweb/.

Keywords: Skill-Imitation, Quadrupedal Robots, Imitation Learning



a) Pacing b) Trotting c) Hopping d) Bounding

Figure 1: SDS's ability to imitate demonstrated skills (top), in simulation (center), and real-world (bottom). The blue tape corresponds to the rear legs, and the red tape to the left-side legs. The a)-d) ordering of the skills is assumed to be consistent throughout the paper.

1 INTRODUCTION

Recent advances in large-scale foundational models—particularly language models (LLMs) [1, 2] and visual-language models (VLMs) [3, 4]—have unlocked new capabilities in robotics, enabling perception-action alignment and goal-driven reasoning [5, 6]. While progress has largely focused on manipulation [7, 8], quadrupedal locomotion remains a uniquely challenging domain due to its high-dimensional, contact-rich, and morphology-dependent dynamics [9]. Acquiring diverse and

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

robust locomotion gaits typically requires costly motion capture [10, 11], heavily engineered reward functions [12, 13], or manually tuned phase-scheduled controllers [14], making current pipelines brittle, non-scalable, and limited in skill diversity. While periodic in nature, each gait generation requires precise, morphology- specific tuning of phase offsets, contact timing, and stabilization gains to differentiate across gaits and embodiments

To address this, we propose **SDS** ("See it. Do it. Sorted."), an automated framework that learns diverse quadrupedal gaits directly from a single video, without MoCap, handcrafted rewards, or gait schedules. SDS leverages GPT-4o to synthesize executable reward functions ($\mathcal{RF}$) from video demonstrations via a novel multimodal prompting pipeline. Specifically, we introduce a grid-based video prompt (G_v) that adaptively samples motion-rich frames, and a chain-of-thought strategy (SUS) that decomposes high-level behaviors into semantically structured subgoals, enabling the VLM to generate compact, skill-differentiable reward terms. These rewards drive PPO-based policy learning in simulation and evolve through closed-loop visual feedback without predefined fitness metrics [15]. SDS explicitly targets periodic locomotion gaits, which remain an open challenge due to their morphology-specific coordination demands. Unlike existing pipelines requiring dense reward shaping and manual retuning, SDS produces transferable, class-separable gaits using only 2–6 reward terms and achieves zero-shot deployment across embodiments. Our key contributions are: **(a)** SDS, a general framework for quadrupedal gait learning from video; **(b)** a multimodal prompting system for semantically grounded reward synthesis; **(c)** sim-to-real deployment of four diverse gaits on a Unitree Go1 robot and (d) Cross-embodiment skills transfer to ANYmal and preliminarily to humanoids.

2 RELATED WORK

Learning from expert demonstrations for the imitation of gaits has been explored extensively by the robotics community [16, 7, 17, 18]. In the past, enabling robots to achieve locomotion required extensive reward engineering [13]. This is a challenging and time-consuming process, due to the sensitivity of deep learning algorithms to hyperparameters and weights, which requires trial-and-error fine-tuning, hence making it hard to develop even a single robot gait [19].

As a response, robot control solutions shifted towards the dependence on domain experts and the creation of high-level action plans through learning-based approaches [20, 21]. MoCap systems are commonly used to extract expert data from animal movement in the form of keypoint locations (e.g., joints and base movement). Deep RL then maps these trajectories into low-level policies that replicate agile behaviors [10, 11, 22]. Although these methods outperform classical approaches by eliminating explicit environment modeling [23], they often suffer from reduced generalization and transferability to novel tasks and robotic platforms. Notable contributions leverage information from videos [24], by developing a spatio-temporal re-targeting framework that aligns the spatial trajectories and temporal dynamics of the source MoCap-recorded motion with the robot's capabilities and optimizes them using Reinforcement Learning (RL). Physics-informed 3D reconstruction from monocular videos further enhances motion learning [25]. This method extracts keypoint trajectories, followed by offline trajectory optimization using contact-implicit constraints in a Model Predictive Controller (MPC). However, MPC's high computational cost and real-time optimization demands can introduce latency and require significant computational power.

Recent advances in generative AI have enabled more intuitive task description in robotics. Generative Adversarial Networks (GANs) have been explored to guide agent learning by dynamically selecting reference motions from expert datasets [26, 27, 28]. However, mode collapse remains a key limitation due to the delicate balance between the generator and discriminator networks. Alternatively, [15] employs LLMs trained on code to automate the generation of RL reward functions, encoding task specifications for interpretation and refinement. However, the approach relies heavily on natural language descriptions and still requires manual, task-specific adjustments. We believe that "a picture is worth a thousand words", with such visual approaches introduced in [8, 29], where a Vision-Language Model (VLM) encodes visual and textual data in a shared embedding space,

enabling one-shot robot policy learning. This eliminates manual reward engineering, though its generalization is currently limited to manipulator-based tasks and does not extend to mobile robots. Rocamonde et al. [30] further use VLMs as zero-shot reward models for RL, providing natural language prompts to learn complex behaviors without manually defined rewards. However, the approach faces challenges such as limited spatial reasoning and sensitivity to visual realism in some environments.

The SDS algorithm is a novel automated pipeline that eliminates the substantial human overhead associated with gait creation, by autonomously generating a set of RL reward functions ($\mathcal{RF}\int$) directly from a video. It extends prior reward evolution methods [15] by removing the need for predefined fitness functions and MoCap data. Defining task fitness from video alone is challenging due to missing keypoints, spatial structure, and temporal cues, causing VLM-generated rewards to often miss critical behavioral aspects [30], eg., optimizing only forward motion may ignore gait-specific dynamics essential for imitation. SDS addresses this via a novel prompting strategy and a dynamic evaluation pipeline that iteratively refines $\mathcal{RF}$ using training feedback. We leverage GPT-4o, a vision-language model trained across diverse platforms, to improve generalization and enable fully autonomous reward generation, policy learning, and evaluation from a single video.

3 METHOD

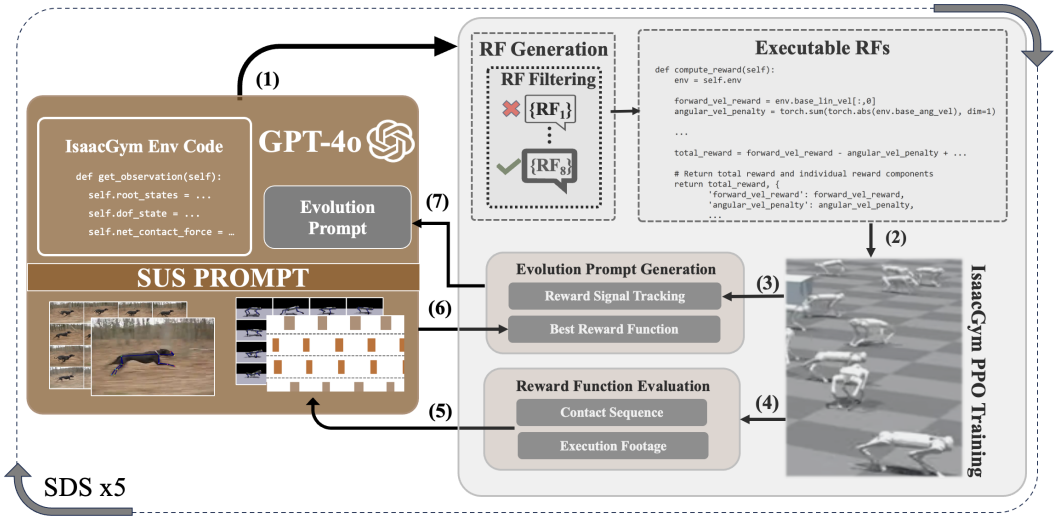


Figure 2: **SDS Method Overview.** The process begins by segmenting the demonstration video into G_v and generating the SUS prompt. Each SDS iteration proceeds as follows: (1) GPT-4o generates a set $\mathcal{RF}$ of 8 candidate reward functions $\mathcal{RF} = \{RF_i\}_{i=1}^{8}$; (2) each executable RF_i is used to train a PPO policy in IsaacGym; (3) sub-reward signals are monitored during training; (4) contact patterns (CP) and rollout footage (G_s) are recorded; (5) RF_i performance is evaluated using the SUS prompt; (6) the best-performing reward RF^* is selected; and (7) RF^* is used to evolve the next iteration's $\mathcal{RF}$.

SDS proposes a novel solution to autonomously generate executable RL python reward functions ($\mathcal{RF}$), for quadrupedal robot skill learning, driven by a single video demonstration input. GPT-4o is selected as the VLM for its strong multimodal reasoning and generalization capabilities, with details in Ap. A.1. Our method introduces novel prompting techniques and a structured task fitness evaluation pipeline to enable the formulation of high-quality $\mathcal{RF}$ for high-fidelity skill imitation.

3.1 Prompting Techniques

We develop novel prompting techniques for video input processing and high-fidelity task decomposition.

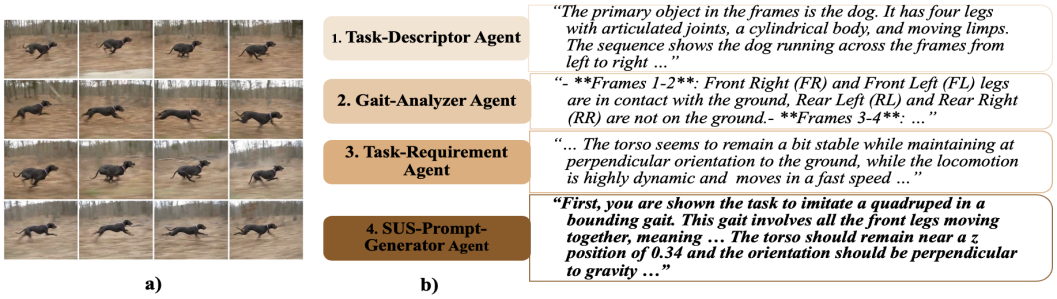


Figure 3: SDS Prompting Techniques for GPT-4o: a) Demonstration video frames, arranged in a grid ($\mathcal{G}_v$) b) SUS agents decomposing $\mathcal{G}_v$ frames into four task-specific prompts

Grid-Frame Prompting: GPT-4o lacks native video understanding and often struggles with temporal coherence. To efficiently standardize and encode video demonstrations for $\mathcal{RF}$ generation, we introduce a grid-based prompting method. Given a demonstration video $\mathcal{V} = \{I_1, I_2, \ldots, I_T\}$ of duration T, we sample n frames adaptively based on the velocity v of the quadruped, with interval τ:

$$n = \frac{T}{v}, \quad \tau = \frac{T}{n}, \tag{2}$$

Lower velocities produce denser sampling to capture finer motion details [31]. The frames are arranged into a spatially uniform grid. This strategy preserves temporal consistency, reduces token budget, and enables parallel visual processing. To mitigate hallucinations and misclassifications (e.g., GPT-4o often misclassifying a quadruped as a floating bench), we augment grid inputs with ViTPose++ keypoints [32], providing a structured motion context, as shown in Fig. 4. This augmentation improves GPT-4o prompting by mapping reward generation to the actual movement of the robot, reducing the ambiguity in the imitation of skills. The uniform frames, augmented with ViTPose++, form the prompting grid $\mathcal{G}_v \in \mathbb{R}^{h \times w}$, where $h = w = \sqrt{n}$.

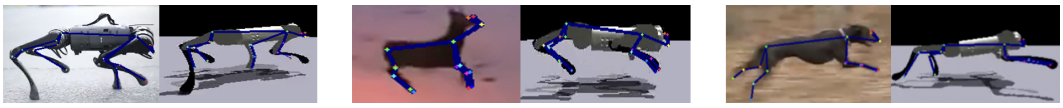


Figure 4: ViTPose++ Estimation on the demonstration frame and corresponding simulation frame.

SUS Prompting: To ensure that the generated $\mathcal{RF}$ aligns with the demonstrated task, we introduce SUS ("See it. Understand it. Sorted."), a multistage prompting technique (Fig. 3.b). Inspired by cognitive reasoning [33, 34], SUS aims to structure the information flow and improve interpretability and decision making, with prompts listed at Ap. A.2.3.a. Given a $\mathcal{G}_v$, SUS uses a multi-agent framework to decompose the complex visual information into 4 task-specific GPT-4o agents, by modifying each agent's system prompts. The *a) Task-Descriptor Agent* is prompted to describe the most likely task being demonstrated. This information is transmitted to *b) Gait-Analyzer Agent*, which analyzes the contact sequence and possible regular gait patterns. Next, *c) Task-Requirement Agent* identifies additional key task characteristics to replicate the demonstration successfully. Finally, *d) SUS-Prompt-Generator Agent* compiles all the information gathered to generate the final SUS prompt, used for the sampling of $\mathcal{RF}$.

3.2 SDS Training Pipeline:

The SDS training pipeline is structured into 7 steps, corresponding to the numbers in Fig. 2:

(1) Reward Function ($\mathcal{RF}$) Generation: We frame $\mathcal{RF}$ generation as a conditional code synthesis problem: GPT-4o is prompted to produce a set of Python reward functions $\mathcal{RF}$, conditioned on $\mathcal{G}v$, the SUS prompt, and environment-specific code (see Fig.2(1)). While SDS is simulator-agnostic, we adopt NVIDIA IsaacGym[35] for direct comparison with prior work [36]. IsaacGym provides

access to environment observations, including base pose, velocities, joint states, and foot contact flags, forming the input space for $\mathcal{RF}$. At each SDS iteration, after prompting GPT-4o outputs a set $\mathcal{RF} = \{RF_i\}_{i=1}^{n}$, where each RF_i is a dictionary containing individual sub-rewards and the aggregated total reward. Sub-rewards are computed over environment variables such as Boolean foot contacts ($\mathbb{R}^4$), joint angles and velocities ($\mathbb{R}^{12}$), and task-specific quantities derived from reference motions. $\mathcal{RF}$ generation is conditioned using a structured system prompt specifying the expected code format, input observations, and reward structure (using Ap. A.2.3.b), together with a user prompt that injects video-specific and task-level objectives (using Ap. A.2.3.c). All RF_i candidates undergo static and runtime validation, ensuring correct Python syntax, API compatibility, and dynamic importability, with invalid samples being discarded (using A.2.3.d). The set of valid functions $\mathcal{RF}_{\text{valid}} \subseteq \mathcal{RF}$ initializes policy training. After empirical tuning, we set $n = 8$ to balance candidate diversity with the risk of hallucinated or non-executable code.

(2)-(4) Isaac Gym PPO Training: Each $RF_i \in \mathcal{RF}_{\text{valid}}$ defines an independent PPO training run within IsaacGym- corresponding to an independent policy, training 4000 parallel quadruped agents per environment. Feasibility is enforced via PhysX, 25 Nm torque clipping, and normalized rewards. Policies are optimized over 1000 iterations, producing an $\mathbb{R}^{12}$ action vector corresponding to target joint positions. Scalar sub-reward components returned by RF_i are logged every 100 iterations to track their contribution and temporal dynamics. Reward components showing unbounded growth are adaptively rescaled, while zero-gradient terms are flagged as uninformative and may be discarded in future iterations, ensuring that only meaningful sub-rewards are retained.

(5)-(6) Reward Function Evaluation: At each SDS iteration, the set $\mathcal{RF}_{\text{valid}}$ is evaluated post-training to identify the best-performing candidate RF^* out of all the RF_i policies. For each $RF_i \in \mathcal{RF}_{\text{valid}}$, the associated policy is deployed in simulation for a 1000-timestep rollout. The inferred robot gait is augmented with ViTPose++ to extract dense keypoint trajectories, rendered into a temporally ordered image grid $\mathcal{G}_s$ (see Ap. A.2.1, Fig. 8), following the same procedure as in Step (1). In parallel, the robot's binary contact sequence $\mathbb{R}^{4\times1000}$ is recorded, capturing per-timestep foot-ground contact states (using Ap. A.2.3.d) and rendered as a contact plot (CP) visualizing inter-limb coordination over time. The combination of pose tracking and CP produces a multimodal behavioral trace that is richer and more interpretable than scalar rewards alone. For evaluation, a structured chat-style prompt is constructed comprising from $\mathcal{G}_s$, CP, and $\mathcal{G}_v$. These elements are embedded into a unified query, paired with a fixed system message template (A.2.3.f), which instructs GPT-4o to assess behavior according to task-specific criteria including *postural stability*, *gait periodicity*, and *trajectory adherence*. GPT-4o returns a score vector in the range 0–10, e.g., $[7, 8, 9]$, each corresponding to a respective metrics. The score vector is parsed into numerical form, summed, and used as the rollout's aggregate performance metric. The RF_i associated with the highest aggregate score is selected as the RF^* for the next iteration. SDS with GPT-4o's multimodal reasoning, replaces brittle hand-crafted rewards with behavior-aligned ones.

(7) Reward Function Evolution: Following evaluation, RF^* is used to seed the next generation of $\mathcal{RF}$ candidates for the subsequent SDS iteration. Its refinement is guided by a structured prompt update that incorporates both the RF^* code and feedback extracted from PPO training logs, in the form of scalar statistics for each sub-reward component converted into natural language summaries (using Ap. A.2.3.g). If training succeeds, positive reinforcement prompts are added; otherwise, traceback diagnostics are included to address execution errors (using Ap. A.2.3.h–i). This feedback, along with $\mathcal{G}_s$, CP, $\mathcal{G}_v$—is compiled into an updated user message. The system prompt (using Ap. A.2.3.b) is reused to form a conversational context that conditions GPT-4o on both the prior RF^* and its empirical performance and generates the new $\mathcal{RF}s$ set for the next SDS iteration. This iterative process enables gradient-free reward optimization via closed-loop GPT-4o interaction, progressively improving reward function quality through behavior-grounded and simulation-informed refinement.

4 EXPERIMENTS & RESULTS

SDS is evaluated on 4 visually similar quadrupedal gaits of increasing dynamic complexity, providing a strong benchmark for imitation fidelity, with task details presented in Tab. 1. For each skill, we run 5 SDS iterations using the learning parameters listed in Ap. A.3, training within 1 day using an NVIDIA RTX-4090 GPU. We transfer our SDS policy running at 50 Hz zero-shot onto a Unitree Go1. The size of $\mathcal{G}_v$ for each skill is defined according to Eq. 2, where the quadruped's velocity is estimated by averaging ViTPose++ keypoints displacement across frames. The resulting $\mathcal{G}_v$ are depicted in Ap. A.2.1. The emergent sub-reward components of the final RF^* used to train each skill policy are detailed in Ap. A.4.1, with specific sub-reward scores provided in Ap. A.4.2.

Skill	**a) Description**	**b) RF* (Ap. A.4.1)**	**c) Vel (m/s)**	**d) $\mathcal{G}_v$**
Pace [37]	Sync movement of adjacent limbs.	Vel, BH, Or, CP, AS, DoFL	0.2	$\mathbb{R}^{6\times6}$
Trot [38]	Sync movement of diagonal limbs.	Vel, BH, Or, LS, AS, DoFL	0.5	$\mathbb{R}^{4\times4}$
Hop [39]	Sync movement of all limbs.	FM, DoFL	1.2	$\mathbb{R}^{4\times4}$
Bound [40]	Sync movement of front limbs.	FM, BH, Or	2.1	$\mathbb{R}^{4\times4}$

Table 1: Overview of the demonstration skills, showing specific behaviors, emergent RF^* sub-reward components, nominal velocities, and $\mathcal{G}_v$ sizes. Acronyms: Vel–Velocity, FM–Forward Motion, BH–Base Height, Or–Orientation, CP–Contact Pattern, LS–Limb Synchronization, AS–Action Smoothness, and DoFL–Degree of Freedom Limits.

4.1 Skill Learning Evaluation

To validate SDS's skill imitation capabilities, we evaluate visual correspondence, contact sequence similarity, and locomotion stability of the learned policies, recording data from ten 1-minute runs. Evaluation is conducted both in simulation and on hardware, with imitation results shown in Fig. 1.

Task Imitation Evaluation:

	a) DTW (10^{-6})				**b) Contact Sequences (%)**			
Skill	sGo1	rGo1	**Avg**$_{go1}$	sANYmal	sGo1	rGo1	**Avg**$_{go1}$	sANYmal
Pace	1.92	2.05	1.99	-	100	100	100	-
Trot	1.28	2.44	1.86	151.49	100	100	100	100
Hop	1.47	2.56	2.01	-	100	100	100	-
Bound	2.85	3.21	2.85	162.54	100	96	98	91.3

Table 2: Average DTW distances and contact sequence matching between demonstration and learned skills over 1-min trial $\times 10$ runs (s-simulation, r-real-world).

Dynamic Time Warping (DTW) [41] is employed to quantify frame-wise similarity between the demonstration footage and the SDS-learned policy, compensating for temporal misalignment by adapting time indices to minimize trajectory distance. Keypoint sequences extracted through ViTPose++, are spatially aligned via Iterative Closest Point (ICP [42] to correct global positional offsets. DTW analysis is conducted across all skills, with results presented in Table 2.a. All skills demonstrate strong trajectory correspondence, with average trajectory distance values on the order of 10^{-6}. A higher DTW value is observed for the bounding skill, primarily due to tracking errors caused by motion blur during high-speed hind limb movements.

Gait Imitation Evaluation: We further assess SDS's gait imitation fidelity (Fig. 5). Simulated contact sequences are encoded as Boolean foot-ground contact states over time, while real-world contact data are extracted from onboard force sensors and smoothed using a moving average filter [43]. Average contact profiles are visualized in Fig. 5, and percent gait matching is reported in Table 2.b. Distinct gait patterns are evident across skills, with both simulated and real-world plots aligning with expected locomotion behaviors (Tab. 1). All skills achieve 100% contact sequence matching to the demonstrated pattern over 1-minute evaluation runs.

Locomotion Stability Evaluation: Real-world gait stability was evaluated by tracking base height fluctuations using both Phasespace [44] motion capture (Fig. 5.b) and OpenCV-based object tracking

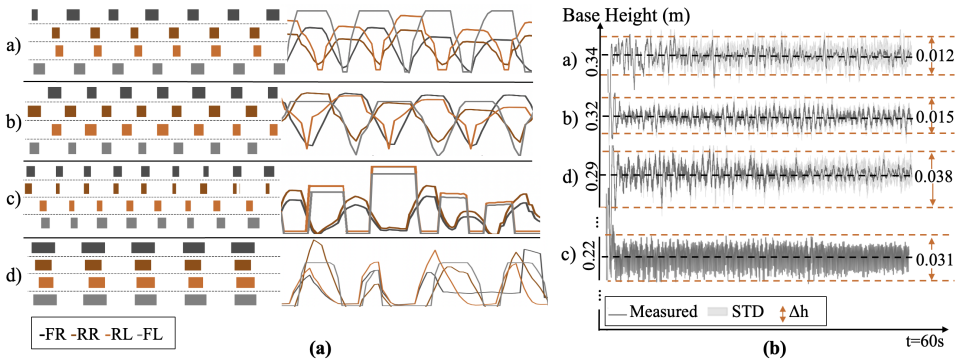


Figure 5: Gait evaluation results. (a) Contact sequences from simulation (right) and real-world smoothed force sensor readings (left) (F: Front, R: Rear; R: Right, L: Left). (b) Average base height and height fluctuation of the real robot in meters. (Note: graph not to scale.)

(Ap. A.7). Across all skills, base height oscillations exhibited a low variance: pacing and trotting exhibited fluctuation variances of 3.6 and 5.6×10^{-5} m, respectively, while bounding and hopping showed slightly larger fluctuations of 2.2 and 3.6×10^{-4} m due to their dynamic nature. This demonstrates the stable control and robustness of SDS. We further compute the Stability Score (StS) by measuring the shortest distance from the robot's CoM to the support polygon and the average angular velocity magnitude across roll, pitch, and yaw. We extended with 50–110 N lateral pushes applied for 2 s at random intervals with the analysis on Ap. A.4.3. The overall average StS across all skills and perturbations is very high at 1.77, given that the maximum possible StS is 2. . Robustness was further validated by increasing ground friction via coarse socks, yielding no significant change (Wilcoxon test, $p = 0.07$).

4.2 Learning Generalization & Component Ablations

Ablations were conducted on the proposed prompting techniques ($\mathcal{G}v$, SUS) and the $\mathcal{RF}$ evolution mechanism components ($\mathcal{G}s$, CP), to evaluate their contributions to the method's performance.

$\mathcal{G}_v$ and SUS Prompting: Removing either prompting mechanism results in complete task learning failure (Fig. 6.a), with 0% contact pattern alignment and DTW distances exceeding $\mathbf{150}$. These results underscore the necessity of both SUS and $\mathcal{G}_v$ prompting for scalable and temporally consistent skill acquisition. Without SUS, $\mathcal{RF}$ components fail to evolve meaningfully, remaining near-static across iterations and preventing task-specific adaptation. Without $\mathcal{G}v$, SDS must process frames independently, increasing computational cost by $\times\mathbf{16}$ due to slower convergence. Moreover, imitation fails because sequential processing lacks temporal coherence, which $\mathcal{G}v$ enforces by enabling spatially and temporally consistent reasoning across frames.

$\mathcal{RF}$ Evolution: Ablating both $\mathcal{G}_s$ and CP collapses task fidelity, reducing contact sequence matching to 0% and driving DTW distances beyond 100, which results in complete imitation and deployment failure. Ablating either component alone also degrades performance: without CP, which enforces local foot contact patterns, structured placements break down, leading to drift and instability and reducing sequence matching to 19%. Removing $\mathcal{G}_s$ has the greatest impact, as it provides global trajectory and motion structure critical for imitation. Its absence eliminates qualitative validation, produces unnatural motions, and destroys overall motion fidelity even when some contacts remain correct, lowering matching to just 4%. Analysis of the final RF^* (Table 1(b)) of each skill further reveals skill-specific dependencies (Fig.6(a)): Pace and Trot, which demand precise limb synchronization, rely strongly on the LS and CP sub-reward components, with ablations causing significant reward degradation. Hop and Bound are less sensitive, relying more on the FM and $DoFL$ sub-rewards. Finally, the proposed evolution framework is validated by the progressive alignment of $\mathcal{RF}$ with task behavior across SDS iterations (1–5), visualized for Trot in Fig.6(b) and across all skills in Ap. A.4.4.

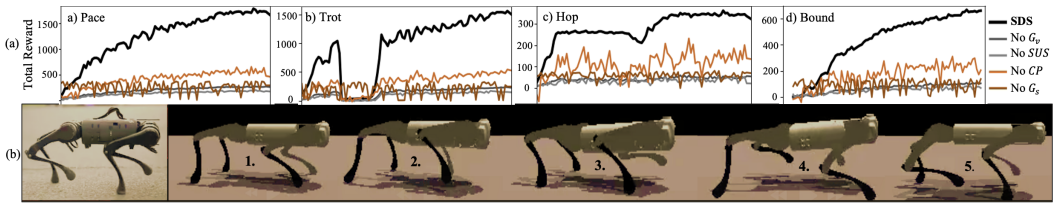


Figure 6: (a) Mean SDS reward signal and ablated variants, across 4 skills trained on Unitree Go1. (b) Evolution of trotting behavior at a matched gait phase (T=5s) across the 5 SDS reward iterations.

Cross-Embodiment: We evaluate SDS on the simulated ANYbotics ANYmal-D [45], a quadruped that differs from the Unitree Go1 by being 38 kg heavier, 25 cm wider, 49 cm taller, and featuring **inverted rear knee joints**. ANYmal-D is trained on trotting and bounding skills to cover both less and more dynamic gaits. Despite significant morphological differences, SDS achieves strong imitation performance, with an average DTW score of 157×10^{-6}, slightly higher than Go1 due to inverted joint keypoints, and a 95% average contact sequence match. These results demonstrate successful skill transfer and generalization, with imitation pairs shown in Ap. 11. We also attempted applying SDS on a Unitree G1 humanoid, with preliminary results achieving lower-limb imitation.

4.3 SDS compared with SOTA methods

We compare SDS with works of similar scope [15, 36, 8, 25, 30], with a summary presented in Table 3 and evaluation metrics detailed in Ap. A.6. Unlike [15], SDS eliminates the need for manual task fitness design and enhances interpretability through the introduction of visual prompting. Compared to [8], SDS achieves superior quadrupedal skill imitation; the method in [8] failed to reproduce demonstrations even after three additional training days, due to its reliance on a task-specific VLM designed for manipulation. SDS also outperforms [25], which requires eight RTX 3080 GPUs and eight hours of optimization per one-minute demonstration, along with a continuous workstation connection. In contrast, SDS runs fully onboard and in real time. Overall, SDS requires no human intervention, replicates motion directly from video, trains efficiently, generalizes through a general-purpose VLM, and supports fully onboard execution across all skills.

Method/Metrics	Train Time	DTW	No Human	No Input Extras	Real World
Eureka [15]	1 day	∞	✗	✗	✗
DrEureka [36]	1 day	>10	✓	✗	✓
RoboCLIP [8]	4 days	∞	✓	✗	✗
SloMo [25]	8 days	>10	✗	✗	✓
VLM-RM [30]	3 days	∞	✓	✗	✗
SDS (Ours)	1 day	1.3×10^{-6}	✓	✓	✓

Table 3: Comparison of SDS with SOTA methods. Metrics include: Train Time for a single skill using a RTX-4090, DTW imitation accuracy for trotting skill, need for human intervention in reward design, extra task inputs, and real-world validation. More detail on the metrics in Ap. A.6

5 CONCLUSION

We present SDS, a pipeline for learning quadrupedal skills from a single demonstration video. SDS leverages GPT-4 to automatically generate reward functions ($\mathcal{RF}$), enabling PPO training in Isaac-Gym without manual reward engineering. Our key contributions include a novel prompting strategy and an autonomous $\mathcal{RF}$ evaluation and evolution framework, allowing precise capture of skill-specific dynamics. SDS was validated through extensive real-world experiments, achieving robust skill imitation across 4 skills with zero resets and low-variance base height oscillations. Quantitatively, SDS attained $> 96\%$ contact sequence matching with demonstrations and maintained DTW distance in the scale of 10^{-6} across all tasks. Compared to SOTA methods, SDS eliminates the need for manual fitness design while increasing fidelity, operates fully onboard and reduces training time.

6 LIMITATIONS & FUTURE WORK

While SDS shows promising results in the imitation of single-video skills, we acknowledge several key limitations. The system currently relies on side-view demonstrations with clear limb visibility, limiting generalization to arbitrary viewpoints and occluded settings. Limited keypoint visibility in videos of short-legged or furry animals (e.g., Persian cats) led to reduced tracking accuracy and poorer imitation quality. We also observe lower imitation success rates for static or less dynamic skills, such as a horse rearing onto two legs. This may be due to limited temporal variation in the demonstration, reduced reliability of contact-based cues, and reduced feedback signal during PPO training. SDS is also validated only on flat terrain and implicitly assumes environmental similarity between simulation and demonstration. However, enhanced real-world deployment demands robustness to diverse terrain properties and domain shifts, which we plan to address through terrain-aware reward generation and domain randomization. Additionally, SDS assumes access to the full state of the simulator, which is not always feasible in real-world conditions; we intend to explore learning from onboard sensing alone using recurrent or belief-based policies. The current single-skill-per-SDS training approach restricts scalability, motivating ongoing work on hierarchical and skill-conditioned architectures for multi-skill integration. Further, SDS currently focuses on periodic locomotion skills, using contact sequences as a lightweight, interpretable feedback signal. We recognize that extending to non-periodic behaviors will require temporally grounded visual inputs (e.g., keypoint trajectories), modified SUS prompting, and new evaluation metrics. Finally, while SDS generalizes to a morphologically distinct quadruped (ANYmal), its effectiveness on platforms with entirely different morphology, such as humanoids, remains untested. We aim to expand SDS to broader morphologies and to analyze the impact of structure and actuation on learning dynamics. **Future work** aims to validate real-world policies in outdoor and unstructured environments, and to extend the skill set to include key locomotion behaviors such as climbing stairs, twisting, and other complex maneuvers. We then aim to combine all skills to enable multi-skill control within a single policy for adaptable quadruped locomotion across diverse environments. In the longer term, we plan to extend SDS to other mobile platforms, with a primary focus on humanoids, which exhibit higher degrees of freedom and pose greater control challenges. We also plan to extend SDS toward non-periodic and trajectory-conditioned skills using temporally-aware rewards (e.g., DTW, trajectory curvature) and integrate high-level modules - e.g., planning, obstacle avoidance [46, 47, 48], while maintaining SDS's focus on observable, deployable behaviors relevant to real-world quadrupedal autonomy.

References

[1] A. Dubey and et al. The llama 3 herd of models, 2024.

[2] OpenAI and et al. Gpt-4 technical report, 2024.

[3] J. Lee, J. Kim, H. Shon, B. Kim, S. H. Kim, H. Lee, and J. Kim. UniCLIP: Unified Framework for Contrastive Language-Image Pre-training, 2022.

[4] S. Geng, J. Yuan, Y. Tian, Y. Chen, and Y. Zhang. HiCLIP: Contrastive Language-Image Pretraining with Hierarchy-aware Attention, 2023.

[5] M. Stamatopoulou, J. Liu, and D. Kanoulas. Dippest: Diffusion-based path planner for synthesizing trajectories applied on quadruped robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[6] J. Liu, M. Stamatopoulou, and D. Kanoulas. Dipper: Diffusion-based 2d path planner applied on legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9264–9270. IEEE, 2024.

[7] C. Wang and et al. Mimicplay: Long-horizon imitation learning by watching human play. In *Conference on Robot Learning (CoRL)*, 2023.

[8] S. A. Sontakke and et al. RoboCLIP: One demonstration is enough to learn robot policies. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[9] N. Kottege, L. Sentis, and D. Kanoulas. Editorial: Towards real-world deployment of legged robots. *Frontiers in Robotics and AI*, Volume 8 - 2021, 2022. ISSN 2296-9144. doi:10.3389/frobt.2021.829403. URL https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2021.829403.

[10] X. B. Peng and et al. Learning agile robotic locomotion skills by imitating animals. In *Robotics: Science and Systems*, 07 2020. doi:10.15607/RSS.2020.XVI.064.

[11] X. B. Peng and et al. Physics-based motion capture imitation with deep reinforcement learning. *Nature*, 2020.

[12] S. Booth and et al. The perils of trial-and-error reward design: misdesign through overfitting and invalid task specifications. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 5920–5929, 2023.

[13] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *5th Annual Conference on Robot Learning*, 2021.

[14] J. Lee, J. Hwangbo, and et al. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47), Oct. 2020. ISSN 2470-9476. doi:10.1126/scirobotics.abc5986.

[15] Y. J. Ma and et al. Eureka: Human-Level Reward Design via Coding Large Language Models. *arXiv preprint arXiv: Arxiv-2310.12931*, 2023.

[16] L. X. Shi and et al. Waypoint-based imitation learning for robotic manipulation. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 2195–2209. PMLR, 2023.

[17] E. Valassakis, G. Papagiannis, N. Di Palo, and E. Johns. Demonstrate once, imitate immediately (dome): Learning visual servoing for one-shot imitation learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[18] D. Totsila, K. Chatzilygeroudis, D. Hadjivelichkov, V. Modugno, I. Hatzilygeroudis, and D. Kanoulas. End-to-end stable imitation learning via autonomous neural dynamic policies, 2023. URL https://arxiv.org/abs/2305.12886.

[19] B. Chen and J. Su. Addressing reward engineering for deep reinforcement learning on multi-stage task. In T. Gedeon, K. W. Wong, and M. Lee, editors, *Neural Information Processing*, pages 309–317. Springer International Publishing, 2019. ISBN 978-3-030-36802-9.

[20] L. Yao, V. Modugno, A. M. Delfaki, Y. Liu, D. Stoyanov, and D. Kanoulas. Local navigation among movable obstacles with deep reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[21] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion, 2024.

[22] S. Bohez et al. Imitate and repurpose: Learning reusable robot movement skills from human and animal behaviors. *Nature*, 2022.

[23] D. Kalashnikov and et al. MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale, 2021.

[24] T. Yoon and et al. Spatio-temporal motion retargeting for quadruped robots, 2024.

[25] Zhang and et al. Slomo: A general system for legged robot motion imitation from casual videos. *IEEE Robotics and Automation Letters*, 8(11):7154–7161, 2023. doi:10.1109/LRA.2023.3313937.

[26] X. B. Peng and et al. Amp: Adversarial motion priors for stylized physics-based character control. In *ACM Transactions on Graphics (TOG)*, volume 40, pages 1–14. ACM, 2021.

[27] T. Wang, H. Kasaei, and H. I. Christensen. Dynamic generative adversarial networks for realistic action imitation. *IEEE Robotics and Automation Letters*, 2021.

[28] J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[29] D. Yang and et al. Rank2reward: Learning shaped reward functions from passive video, 2024.

[30] J. R. et al. Vision-language models are zero-shot reward models for reinforcement learning, 2024.

[31] Y. Zhu and S. Newsam. Random temporal skipping for multirate video analysis, 2018.

[32] Y. Xu and et al. Vitpose++: Vision transformer for generic body pose estimation, 2023.

[33] J. Wei et al. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2023.

[34] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa. Large Language Models are Zero-Shot Reasoners, 2023.

[35] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning, 2022. URL https://arxiv.org/abs/2205.02824.

[36] Y. J. M. andet al. Dreureka: Language model guided sim-to-real transfer, 2024.

[37] I. Starnes. Difference between a pace, stepping pace, and smooth gait/saddle rack, 2015. URL https://www.youtube.com/watch?v=SMmcGUSUSA0&t=44s.

[38] U. Robotics. Quadruped robot a1 walk with you to the future, 2020. URL https://www.youtube.com/watch?v=2H3dzZEi-qw.

[39] D. G. for a Morning Skip Across a Beach. Deer goes for a morning skip across a beach, 2016. URL https://www.youtube.com/watch?v=49fz_WKBrXo.

[40] skimoriver01. Great dane running 30mph fast from the side., 2010. URL https://www.youtube.com/watch?v=A8YPHoc0dR0&pp=ygULZG9nIHJ1bm5pbmc%3D.

[41] *Dynamic Time Warping*, pages 69–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-74048-3. doi:10.1007/978-3-540-74048-3_4.

[42] P. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992. ISSN 1939-3539. doi:10.1109/34.121791.

[43] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall, 2010.

[44] P. Inc. Impulse x2e motion capture system, 2025. URL https://www.phasespace.com/x2e-motion-capture/. Accessed: 2025-04-28.

[45] M. Hutter and et al. Anymal - a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 38–44, 2016. doi:10.1109/IROS.2016.7758092.

[46] Q. Zhang, S. Ma, B. Liu, J. Jiao, and D. Kanoulas. Follow everything: A leader-following and obstacle avoidance framework with goal-aware adaptation, 2025. URL https://arxiv.org/abs/2504.19399.

[47] L. Yao and et al. Local path planning among pushable objects based on reinforcement learning. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3062–3068, Oct 2024. doi:10.1109/IROS58592.2024.10802257.

[48] P. Roth, J. Nubert, F. Yang, M. Mittal, and M. Hutter. Viplanner: Visual semantic imperative learning for local navigation, 2023.

[49] F. Bordes and et al. An introduction to vision-language modeling, 2024.

[50] Z. Wang and et al. Language models with image descriptors are strong few-shot video-language learners, 2022.

[51] B. Ni and et al. Expanding Language-Image Pretrained Models for General Video Recognition, 2022.

[52] K. Lin and et al. SwinBERT: End-to-End Transformers with Sparse Attention for Video Captioning, 2022.

[53] B. Lin and et al. Video-llava: Learning united visual representation by alignment before projection, 2023.

[54] Z. Tong and Y. al. VideoMAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training, 2022.

[55] OpenAI. Gpt-4v(ision) system card, 2023.

A Appendix

This section provides additional information to support the main paper.

A.1 VLM Selection:

Vision-Language Models (VLM) encode joint visual-text representations through large-scale multimodal training [49]. Given an input sequence $V = I_1, I_2, ..., I_T$ and textual descriptions T, parameters θ are optimized by masked modeling, contrastive objectives, or causal generation. Standard VLMs, trained primarily on image-text pairs, struggle with temporal reasoning essential for skill learning. Video-based adaptations extend pre-trained encoders $f_\theta(I)$ with temporal mechanisms [50, 51] or adopt end-to-end video pre-training with masked modeling [52] and contrastive learning [4]. We evaluated other SOTA VLMs [53, 54] under identical structured prompts which despite strong VQA performance, they failed to robustly produce executable Python code. We also observed that SOTA pipelines using morphology specific VLMs (e.g., S3D) limit cross-embodiment generalization. GPT-4o(ision) [55] was selected for its ability to synthesize structured, executable Python code from visual prompts and its broad multi-modal training, which is critical for extending SDS beyond quadrupeds. GPT-4o's robust visual reasoning and multi-modal dataset ($\mathcal{D}$) spanning both animal and human data, will aid in improving generalization of SDS. GPT-4o follows a two-stage training:

$$\mathcal{L}_{\text{GPT-4o}} = \mathbb{E}_{(x_t, x_v)\sim\mathcal{D}} \left[\log P(x_t, x_v \mid \theta)\right] \tag{1}$$

where x_t and x_v are textual and visual tokens, respectively. Fine-tuned for domain-specific applications, GPT-4o enables structured task decomposition and robust visual input reasoning, making it well-suited for robot skill learning.

A.2 SDS Prompting

A.2.1 Video Prompting

The grids (G_V) of the demonstration videos used to train the different SDS skills are presented in Fig. 9.

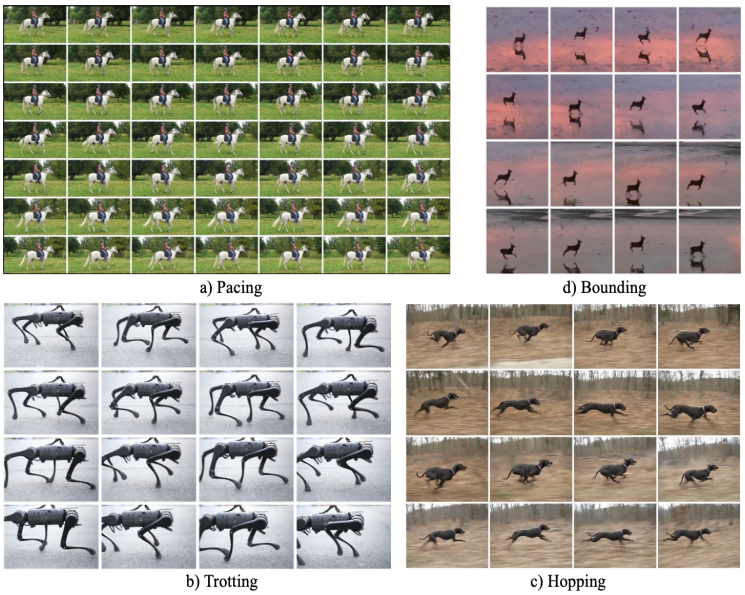


Figure 7: Demonstration videos arranged in a grid formation ($\mathcal{G}_v$), serving as input to GPT-4o for SDS processing.

The simulation footage frames for the Trotting skill across SDS iterations, organized into grids $\mathcal{G}_s$, for GPT-4 prompting, presented in Fig. 8.

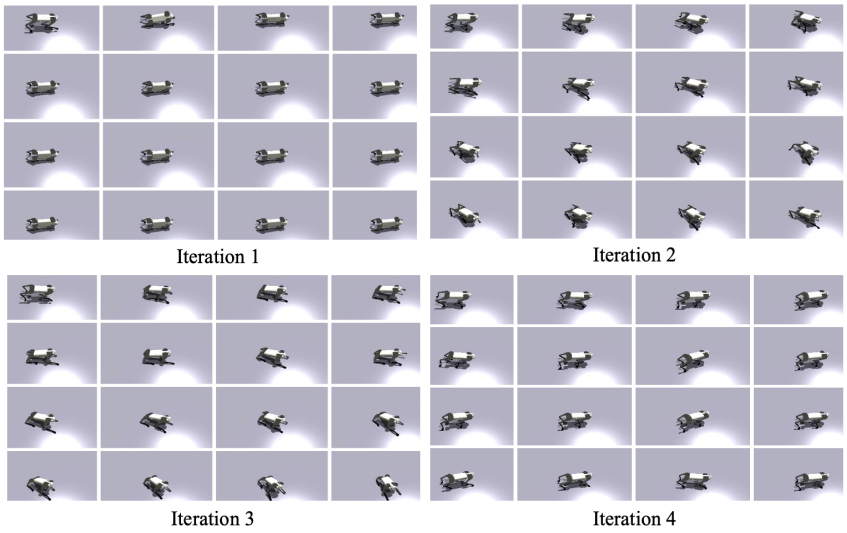


Figure 8: Simulation footage arranged in a grid formation ($\mathcal{G}_s$), serving as input as GPT-4o input.

A.2.2 SUS Prompting

Task decomposition of the trotting demonstration video produced by the four task-specific GPT-4o agents, presented in Fig. 9.

1. Task-Descriptor Agent
"The primary object in the frames is the quadruped robot. It has four legs with articulated joints, a cylindrical body, and various mechanical components. The sequence shows the robot walking across the frames from left to right …"

2. Gait-Analyzer Agent
"- **Frames 1-2**: Front Right (FR) and Rear Left (RL) legs are in contact with the ground, Front Left (FL) and Rear Right (RR) are off the ground.
- **Frames 3-4**: Front Right (FR) and Rear Left (RL) legs propelled backwards, and Front Left (FL) and Rear Right (RR) legs are in contact with the ground…"

3. Task-Requirement Agent
"… The torso seems to remain rather stable while maintaining at perpendicular orientation to the ground, while the locomotion remains at a constant and moderate speed …"

4. SUS-Prompt-Generator Agent
"First, you are shown the task to imitate a quadruped in a trotting gait. This gait involves the same-side pairs of legs moving together, meaning … You must keep the torso should remain near a z position of 0.34 and the orientation should be perpendicular to gravity…"

Figure 9: SUS Prompting:4 Task-Specific GPT-4o Agents decomposing the Trotting task.

A.2.3 Method Prompts

Prompt File	Step	Description
a) `sus_generator, SUS`	(0)	Prompt used to generate the structured SUS (Skill Understanding String) from gait annotations and demonstration metadata; output is inserted into the reward generation prompt.
b) `init_sds_system`	(1)	System prompt instructing GPT-4o to generate executable reward functions compatible with Isaac Gym's observation API. Defines function structure and behavioral objectives.
c) `sds_user`	(1)	User prompt injecting task-specific goals based on gait analysis and demonstration video, guiding reward synthesis toward desired behaviors.
d) `code_output_tip`	(1,7)	Auxiliary prompt appended to enforce complete, syntactically valid code outputs, discouraging truncation or malformed syntax.
e) `contact_sequence_system`	(5)	Supplementary system prompt used when evaluating foot-ground contact sequences, emphasizing gait periodicity and limb coordination.
f) `init_task_evaluator_system`	(5)	System prompt defining multimodal evaluation criteria (stability, periodicity, trajectory fidelity) for GPT-4o-based policy scoring.
g) `policy_feedback`	(7)	Positive feedback prompt included when a training run completes successfully, summarizing reward statistics and learning progress.
h) `code_feedback`	(7)	Positive reinforcement prompt returned when generated reward code executes correctly and produces meaningful learning dynamics.
i) `execution_error_feedback`	(7)	Diagnostic prompt containing traceback and error information, sent to GPT-4o when code fails to compile or execute during validation.

Table 4: Prompt files used in SDS and their corresponding roles within each step of the methodology. Prompt files can be accessed at: https://github.com/sdsreview/SDS_ANONYM/tree/main/SDS/prompts

A.3 Learning Parameters

We present the learning parameters used to train SDS.

LR	Clip	Entropy	Mini-batches	Epochs	Gamma	Max Iters
0.001	0.2	0.01	4	5	0.99	1500

Table 5: PPO training hyperparameters used consistently across all SDS skills.

Parameter	Value
Initialization Noise Std.	1.0
Actor Hidden Layers	[512, 256, 128]
Critic Hidden Layers	[512, 256, 128]
Activation Function	ELU
Adaptation Branch Hidden Dims	[[256, 32]]
Env Factor Encoder Input Dims	[18]
Env Factor Encoder Latent Dims	[18]
Env Factor Encoder Hidden Dims	[[256, 128]]

Table 6: Actor-Critic (AC) network configuration used across SDS skill training.

A.4 Reward Functions

A.4.1 Reward Components Description

Presenting the potential sub-rewards emerging within the RF_i dictionary.

Component	Description	Unit	Dim.
Velocity (Vel)	Encourages matching the commanded base linear velocity.	[m/s]	$\mathbb{R}^{4\times4}$
Forward Motion (FM)	Rewards forward translational motion aligned with the command direction.	[m/s]	$\mathbb{R}^{4\times4}$
Base Height (BH)	Rewards maintaining a target base height to ensure locomotion stability.	[m]	$\mathbb{R}^{1}$
Orientation (Or)	Penalizes deviation from upright orientation using quaternion distance.	[rad]	$\mathbb{R}^{1}$
Contact Pattern (CP)	Encourages limb contact timings to match a desired gait pattern (e.g., pacing).	[binary match]	$\mathbb{R}^{4\times T}$
Limb Sync (LS)	Rewards synchronized limb movement (e.g., diagonal or bounding gaits).	[unitless]	$\mathbb{R}^{1}$
Action Smoothness (AS)	Penalizes abrupt action changes to promote smoother joint torques.	[rad/s]	$\mathbb{R}^{n}$
DoF Limits (DoFL)	Penalizes joint positions near mechanical limits to avoid over-extension.	[rad]	$\mathbb{R}^{n}$

Table 7: Descriptions, units, and dimensions of reward components used across different SDS skills.

A.4.2 RF^* Sub-Reward Scores

We present the sub-reward scores for the final policy of each skill in Table 8, and the aggregate reward sum and failure rate for each skill in Table 9. Total RF^* values for each skill, showing the total aggregate reward value and reset value- occurring when the robot reaches a termination condition (base or joint hit the ground, orientation diverges from limits).

Skill	Vel	FM	BH	Or	CP	LS	AS	DoFL
Pace	0.026	–	11.191	0.001	4.140	–	0.874	11.950
Trot	0.153	–	11.192	0.001	–	7.717	0.875	2.391
Hop	–	1.480	–	–	–	–	–	0.006
Bound	–	0.514	11.574	11.474	–	–	–	–

Table 8: Final RF^* sub-reward component values for each learned skill. Missing values (–) indicate the absence of that reward in the respective skill.

Skill	Total Reward	Reset Events
Pace	3.301	0
Trot	14.821	0
Hop	5.986	0
Bound	9.157	0

Table 9: Total reward and reset events for each learned skill. All policies remained stable, with no reset events during training.

A.4.3 Policy Stability Evaluation

We further evaluate robustness using two stability metrics: the shortest distance from the center of mass to the support polygon (CoM_{dist}) and the average angular velocity magnitude ($|\omega|$) computed over roll, pitch, and yaw. These are combined into a single score, the Stability-to-Speed ratio (StS), defined as:

$$\text{StS} = 2 - [\text{clip}(\text{CoM}_{\text{dist}},\ 0,\ 1) + \text{clip}(|\omega|,\ 0,\ 1)] \tag{2}$$

A higher StS indicates better stability, with a maximum possible score of 2. To further test robustness, we applied lateral perturbation forces ranging from 50N to 110N for 2s at random time intervals. We report StS scores under both unperturbed (0N) and maximum perturbation (110N) conditions.

Table 10: Policy Stability Score (StS) under 0N and 110N lateral perturbations (2s duration).

Gait	StS (0N)	StS (110N)	Skill Mean
Pace	1.86	1.75	1.81
Trot	1.92	1.79	1.86
Hop	1.77	1.64	1.71
Bound	1.78	1.68	1.73
Perturbation Mean	**1.83**	**1.72**	**1.77**

A.4.4 $\mathcal{RF}$ Evolution

The behavioral evolution of the agents across five iterations of the SDS process, as generated by GPT-4o, illustrates the progression of RF evolution. Snapshots were captured at the same phase of the gait cycle ($T = 5$ s) for each iteration, as shown in Fig. 10.

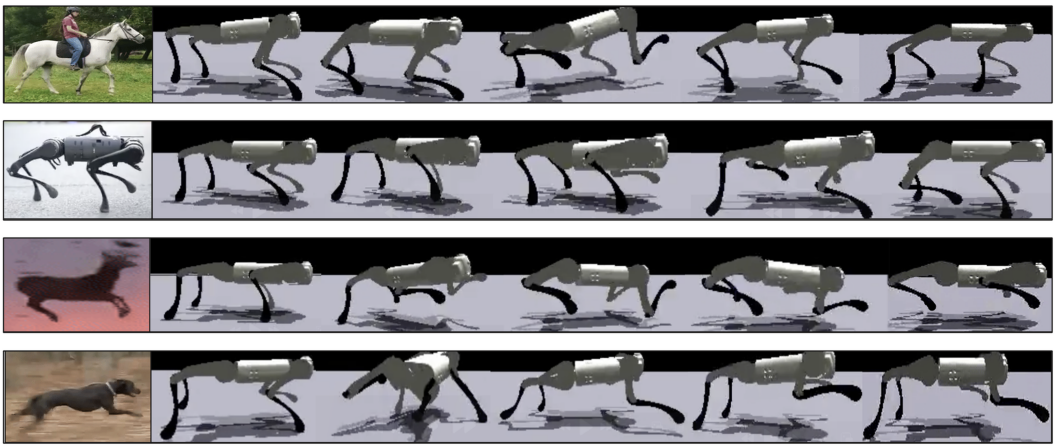


Figure 10: Evolution of task behavior of all skills at a matched gait phase (T=5s) across the 5 SDS reward iterations.

A.5 SDS Generalisation

To assess the generalization of SDS, we train the trotting and bounding skills on ANYmal [45], featuring a different morphology than the one of the demonstrated videos, and therefore altering the kinematic constraints due to its inverted rear knee joints. We chose trotting and bounding as the target skills because they are commonly used locomotion behaviors and vary significantly in their dynamic properties. Example imitation pairs are shown in Figure 11.

a) Trotting b) Bounding

Figure 11: Demonstration of the generalization capabilities of SDS on the ANYmal quadruped robot, featuring an inverted joint configuration and significantly different kinematics. Comparison between demonstration and learned behavior frames for a)trotting and b) bounding skills. (Red corresponds to left-side legs)

A.6 Comparison Metrics

We provide details on the metrics selected to compare SDS with state-of-the-art methods of similar scope in Tabel 11, corresponding to Table 3.

Metric	Description
Train Time	Duration required to train a single skill on an RTX-4090 GPU, providing a measure of computational efficiency.
DTW	Dynamic Time Warping: a frame-wise imitation accuracy metric measuring temporal alignment between predicted and demonstrated trajectories. Lower is better; ∞ denotes poor alignment.
No Human	Whether the method avoids human intervention in reward design, e.g., fine-tuning rewards or fitness functions. A check mark (✓) indicates no human-in-the-loop.
No Input Extras	Whether the method avoids requiring additional task inputs like natural language instructions, segmentation masks, or depth maps. ✓indicates no such inputs are needed.
Real World	Whether the method has been validated in real-world robotic experiments beyond simulation. ✓denotes successful real-world deployment.

Table 11: Description of evaluation metrics used for comparing SDS with state-of-the-art methods.

A.7 Real-World Experiments

Showcasing the base height oscillation trajectory traces of the real Unitree Go1 robot across all skills, in Fig. 12.

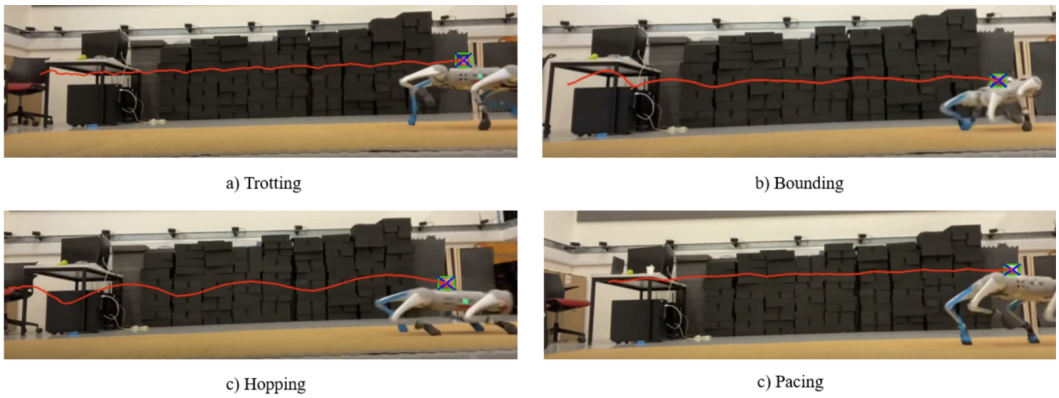


Figure 12: Real Robot Stability: Base Height tracing of real robot, red line indicates the trajectory