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

Jeffrey Li*, Maria Stamatopoulou*, and Dimitrios Kanoulas

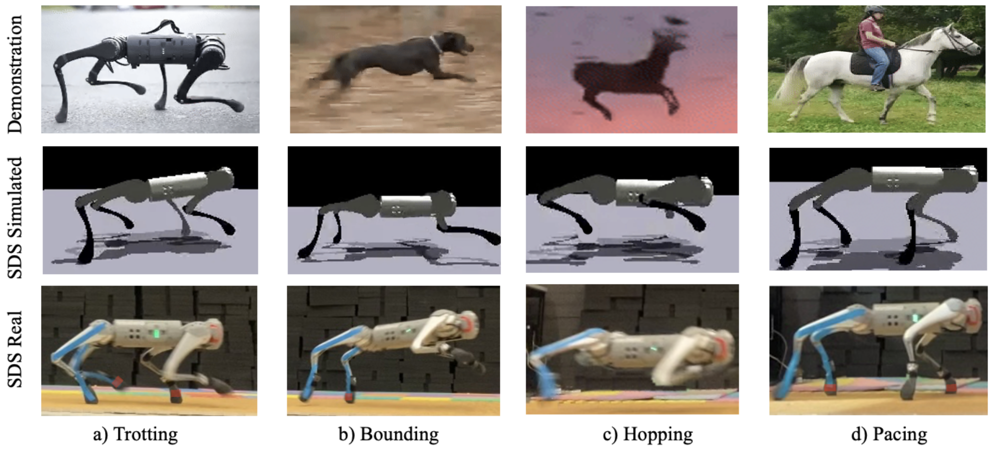


Fig. 1: SDS' ability to imitate demonstrated videos, in simulation and real-world (blue: rare legs, red: left-side legs).

***Abstract*— In this paper, we present SDS ("See it. Do it. Sorted."), a novel pipeline for intuitive quadrupedal skill learning from a single demonstration video. Leveraging the Visual capabilities of GPT-4o, SDS processes input videos through our novel chain-of-thought promoting technique (SUS) and generates executable reward functions (RFs) that drive the imitation of locomotion skills, through learning a Proximal Policy Optimization (PPO)-based Reinforcement Learning (RL) policy, using environment information from the NVIDIA IsaacGym simulator. SDS autonomously evaluates the RFs by monitoring the individual reward components and supplying training footage and fitness metrics back into GPT-4o, which is then prompted to evolve the RFs to achieve higher task fitness at each iteration. We validate our method on the Unitree Go1 robot, demonstrating its ability to execute variable skills such as trotting, bounding, pacing and hopping, achieving high imitation fidelity and locomotion stability. SDS shows improvements over SOTA methods in task adaptability, reduced dependence on domain-specific knowledge, and bypassing the need for labor-intensive reward engineering and large-scale training datasets. Additional information and the open-sourced code can be found in: https://rpl-cs-ucl.github.io/SDSweb.**

I. INTRODUCTION

Over the past decades, advancements in computational hardware have spurred the development of foundational models [1], particularly in the form of Large Language Models (LLMs) [2]–[4] and Visual Language Models (VLMs) [5]–[7]. These models, trained on extensive datasets, demonstrate versatility and can be integrated into robots for real-world applications, such as industrial and agricultural inspections, monitoring of hazardous environments and search and rescue operations.

Replicating animals' diverse and agile locomotion skills has long posed a challenge in robotics. Recent works have explored this through imitation learning, utilizing Motion Capture (MoCap) systems to track and replicate actions [8]–[10], or through visual demonstrations to guide reward function engineering in Reinforcement Learning (RL) [11]–[13]. However, these methods often require extensive manual labeling, specific domain knowledge and vast amounts of training data making even a single skill challenging to learn. As a response, leveraging foundational models to synthesize low-level controllers for robots is emerging as a research focus, especially for quadrupedal robots, which are valued for their ability to navigate complex environments, while maintaining stability [14], [15]. Foundational models can bridge cognitive task understanding and low-level robotic actions [16]–[19] by using image or video demonstrations of animal motor skills, offering a more intuitive approach to task instruction and direct mapping to action generation.

Leveraging the latest advancements in VLMs, we propose an innovative algorithm, named SDS ("See it. Do it. Sorted."), which utilizes GPT-4V(ision) to generate reward functions for learning locomotion skills applied to quadrupedal robots. SDS processes a single video of any quadrupedal agent (animal or robot) performing the desired task and generates reward functions that drive the learning of

The authors are with the RPL Lab, Department of Computer Science, University College London, Gower Street, WC1E 6BT, London, UK. D. Kanoulas is also with the AI Centre, Department of Computer Science, University College London, Gower Street, WC1E 6BT, London, UK and Archimedes/Athena RC, Greece. {jeffrey.li.20, maria.stamatopoulou.21, d.kanoulas}@ucl.ac.uk

*equal contribution

This work was supported by the UKRI Future Leaders Fellowship [MR/V025333/1] (RoboHike). For the purpose of Open Access, the author has applied a CC BY public copyright license to any Author Accepted Manuscript version arising from this submission.

low-level control policies through RL. Our novel approach improves upon the SOTA framework [20], automating and refining the evolutionary search of reward functions with visual input, to better imitate the demonstrated task. SDS represents the development of single-demonstration skill learning, facilitating multi-skill platform-invariant policies, minimizing dataset size requirements and removing the need for manual reward engineering. We also propose SUS ("See it. Understand it. Sorted."), a novel prompting technique that breaks down a task prompt into intuitive sub-steps, guiding the model through the reasoning stage for more accurate and coherent problem-solving. Our contributions are summarized as follows. We introduce:

- High-fidelity skill imitation through single video demonstration using VLMs.
- SUS: Novel chain-of-thought prompting technique
- SDS: Video-to-reward function pipeline for low-level skill learning using RL.
- Real-world deployment stack with experiments using the Go1 robot.

The remainder of the paper is structured as follows: In Sec. II, we review relevant literature on skill imitation. Sec. III provides the necessary background on RL and VLMs, while Sec. IV details our proposed method, with experimental results presented in Sec. V. Finally, Sec. VI summarizes our findings and outlines future work.

II. RELATED WORK

Learning from expert demonstrations for skill imitation has received increased attention from the robotics community [21]–[23]. In the past, enabling robots to achieve locomotion required extensive reward engineering [24]. This is a challenging and time-consuming process, due to the sensitivity of deep learning algorithms to hyper-parameters and the specific weights assigned to different reward components, requiring trial-and-error fine-tuning, making it challenging to develop even for a single robot skill [25].

As a response, robot control solutions shifted towards dependence on domain experts and the creation of high-level action plans through learning-based approaches [26], [27]. This often involves the use of MoCap systems to extract expert data from animal movement in the form of keypoint locations (e.g., joint and base movement), which is then used in combination with Deep RL systems to train low-level policies replicating the behaviors [8]–[10]. While these methods outperform classical approaches by eliminating explicit environment modeling [28], they often suffer from reduced generalization and transferability to novel tasks and robotic platforms. Notable contributions are proposed by leveraging information from videos [29], 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 RL. A physics-informed 3D reconstruction pipeline is proposed in [30] to extract key-point trajectories from monocular videos, followed by offline trajectory optimization with contact-implicit constraints through a Model Predictive Controller (MPC) to generate dynamically feasible robot motion. However, MPC's high computational cost and real-time optimization demands can introduce latency and require significant computational power.

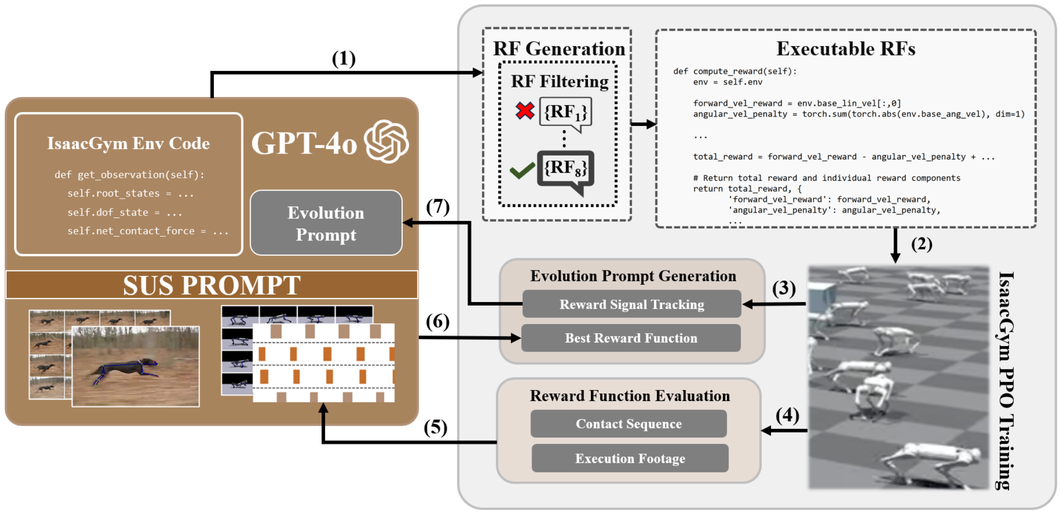
Recent works are moving towards using Generative AI, as it can offer a more intuitive approach to task instruction. One such direction includes leveraging Generative Adversarial Networks (GANs) to drive the agent's learning process towards closely matching the target motion, by dynamically choosing whether the reference motion originates from a random RL agent or an expert dataset [31]–[33]. However, this often suffers from mode collapse issues due to the delicate balance required between the generator and discriminator networks. Another approach [20] uses LLMs trained on extensive programming data to automate the generation of RL reward functions. This is achieved by encoding the desired task specifications for LLM interpretation to produce reward functions integrated into the RL agent's training process and iteratively refined. Although "a picture is worth a thousand words", it highlights the limitations of relying solely on natural language descriptions for task delineation. A visual aspect to skill learning is introduced in [34], combining a VLM and RL to enable one-shot learning of robot policies. The VLM encodes visual and language data into a shared embedding space, which allows the system to map visual observations directly to corresponding actions based on a single demonstration. This provides a starting point for the RL process, which fine-tunes the policy with minimal additional data. This approach allows effective learning of robot skills without any manual reward engineering required, however, the VLM is specifically trained for manipulators and does not generalize to mobile robots. In our solution, we leverage the GPT4-V VLM, due to its generalization capabilities, to enhance upon [20] by enabling single-demonstration videos to autonomously drive all reward generation, skill acquisition, and reward evaluation. Our approach allows the system to interpret and map visual data from a single demonstration directly into reward signals, facilitating the learning of complex skills with minimal data.

III. PRELIMINARIES

A. Reinforcement Learning (RL)

Reinforcement Learning (RL) aims to optimize the expected cumulative rewards that an agent accumulates over time. At each time step t, the agent observes the current system state x_t and selects an action a_t according to the current policy $\pi(a_t|x_t)$, where the action a_t is sampled from this policy. The chosen action transitions the system to a new state x_{t+1}, and the agent receives a reward $r_t = r(x_t, a_t, x_{t+1})$ based on the current state, the action taken, and the resulting new state.

This study uses the Proximal Policy Optimization (PPO) algorithm to discover the optimal policy. PPO is an actor-critic policy gradient method that has demonstrated robust performance in solving complex tasks [35]. PPO is designed to train a stochastic policy using an on-policy learning approach. The PPO loss function is defined as the expectation



```
def get_observation(self):
  self.root_states = ...
  self.dof_state = ...
  self.net_contact_force = ...
```

```
def compute_reward(self):
    env = self.env

    forward_vel_reward = env.base_lin_vel[:,0]
    angular_vel_penalty = torch.sum(torch.abs(env.base_ang_vel), dim=1)

    ...

    total_reward = forward_vel_reward - angular_vel_penalty + ...

    # Return total reward and individual reward components
    return total_reward, {
            'forward_vel_reward': forward_vel_reward,
            'angular_vel_penalty': angular_vel_penalty,
            ...
```

Fig. 3: SDS Training Pipeline: The process begins with gridifying the demonstration video and generating the SUS prompt (Sec. IV-A). This is followed by the SDS iterations involving: (1) Generating 8 python-based reward functions (RFs) through GPT-4o (Sec. IV-B) and filtering them. (2) The executable RFs drive the skill imitation through learning of PPO policies in IsaacGym (Sec. IV-B). (3) The values of each RF's reward components are monitored during training. (4) During the final checkpoint of each RF, Contact Sequence and Execution footage are recorded (Sec. IV-C) (5) Performance of the RFs is evaluated by GPT-4o using the SUS prompt and monitored data from (4) (Sec. IV-D) (6) The locally optimal RF is identified (7) and is then used as a basis for the evolution prompt along with its monitored reward signal to generate evolved RFs, which are queried and used in subsequent training iterations (Sec.IV-E).

over the minimum of the standard policy gradient objective and a clipped version of the objective, which introduces a constraint that limits how much the new policy can deviate from the old policy. The clipped objective is defined as:

$$L^{clip}(\theta) = \hat{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right], \tag{1}$$

where $r_t(\theta)$ denotes the probability ratio of the action under the current policy relative to the previous policy, and $\hat{A}_t$ represents the advantage function, which estimates the relative value of the selected action in the current state by considering the discounted reward minus the value function estimate.

B. VLMs

Visual Language Models (VLMs) are multi-modal foundational models able to process text and images simultaneously. They are trained on web-scale visual-text data and can perform zero-shot inference on unseen data, demonstrating their versatility and robustness in various applications. At their core, VLMs combine a) machine vision, i.e., translating raw pixels into object representations; b) LLMs, which offer interpretability and connect content expressed across different contexts; and c) fusing agents, for the interaction of (a) and (b). VLMs are particularly beneficial in applications such as robotics, as they allow robots to understand and capture concepts that are difficult to describe using natural language alone, such as skill learning.

In this work, we utilized GPT-4V(ision), an extension of GPT-4o developed by OpenAI, that incorporates image input to perform multi-modal learning. GPT-4V utilizes a transformer architecture, undergoing a two-phase training process. In the pre-training phase, it learns to understand and generate text and images by analyzing extensive datasets. Subsequently, it undergoes domain-specific fine-tuning for specific applications. With GPT-4o as its foundation, GPT-4V demonstrates a superior ability to reason about the content of images in conjunction with textual information, surpassing the performance of existing models [36].

Although most research on VLMs has focused on images as the visual input, VLMs trained on videos can offer a better conceptual understanding of skill learning. Training such models can be achieved either by adapting existing image-based language models to handle video input, as seen in models like VidIL [37] and XCLIP [38] or by pre-training a model from scratch which typically employ a BERT-style masked modeling objective [39] or contrastive learning objectives (CLIP).

IV. METHOD

The SDS algorithm proposes a novel solution to generating optimal RL reward functions (RFs), driven by single demonstration video input, for quadrupedal robot skill learning. Leveraging the NVIDIA IsaacGym environment and building upon the reward evolution pipeline established in [20], our method introduces a novel prompting technique and task fitness evaluation pipeline to ensure optimal RF formulation for skill imitation. We run 5 SDS iterations, where each iteration involves the generation of 8 RFs, followed by PPO training and evaluation of the RFs for the quadrupedal agent.

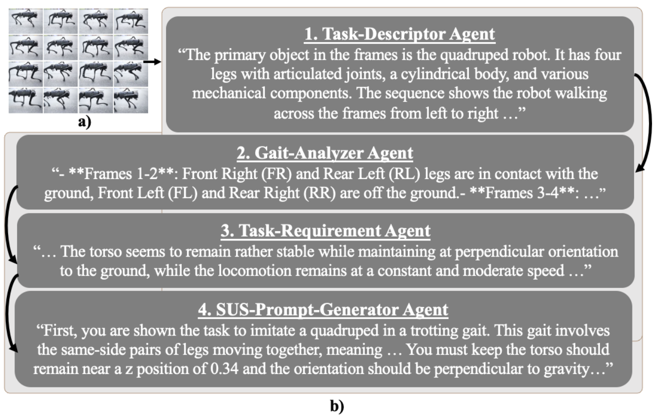
Each of the 8 RFs undergoes $1,000$ PPO training iterations, with the entirety of SDS taking a maximum of 1 day to train, using a single NVIDIA RTX-4090 GPU.

A. Prompting Techniques

We develop novel techniques for video and task prompting.

1) Video Prompting: In choosing a VLM, we tested current SOTA video VLM models [40], [41], that perform well for visual-language tasks (e.g., video captioning), but usually struggle with outputting structured, syntactically and semantically correct Python code.

Hence, we propose a novel video prompting technique using GPT-4V(ison) [36], part of GPT-4o. GPT-4V does not directly process video input, so to enhance the retention of cohesive information while ensuring the generation of high-quality Pythonic RFs, we propose sampling video frames at regular intervals and arranging them uniformly on a grid (Fig. 4.a), with sampling rate = $\sqrt{\text{grid size}}$. The grid size is varied for each task and is selected based on the quality and resolution of the video, on striking the ideal balance of preserving contextual information across frames and minimizing context loss, e.g., videos with lower frame rates would have a larger grid size. Varying the size is important as larger grid dimensions increase computational costs and cause GPT-4V to lose focus on the primary task of generating reward functions. In contrast, smaller grids sacrifice critical contextual information between frames, which is essential during the task fitness evaluation phase. By employing this grid-based encoding, GPT-4V can effectively interpret the demonstrated task and generate appropriate RFs while reducing the overall cost compared to sequentially processing individual frames.


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**: …”

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…”

Fig. 4: SDS Prompting Techniques: a) Gridified demonstration videos b) SUS task decomposition Agents prompting.

2) SUS Prompting: To ensure that the generated RFs do not deviate from the primary task demonstrated in the video, we introduce a multi-stage RF prompting process, namely SUS (”See it. Understand it. Sorted.”) prompting (Fig. 4.b). This approach, inspired by human cognition and [42], [43], leverages a multi-agent system to decompose complex visual information into smaller, more interpretable units, facilitating understanding and decision-making. SUS decomposes the task by creating 4 task-specific GPT-4o Agents, by modifying the system prompts for each conversation. When the gridified frames are inputted in the SUS, the: *a) Task-Descriptor Agent* is prompted to describe and explain the most likely task being demonstrated. This information is then relayed to the *b) Gait-Analyzer Agent*, which analyzes the most probable contact sequence and potential regular gait patterns based on the frames and the task description. Next, the *c) Task-Requirement Agent* is tasked with identifying additional key characteristics necessary to replicate the demonstration successfully. Finally, the *d) SUS-Prompt-Generator Agent* compiles all the gathered information to generate the final SUS prompt. This SUS prompt and the gridified frames are finally used to query for RF sampling. This approach has significantly improved the relevance of the generated reward functions to the specific demonstration video.

B. Reward Function Generation

Querying GPT-4o for RF sampling in our method requires the simulators' environment code, gridified demonstration video representation, and the SUS prompt, with the latter two components obtained from Sec. IV-A. Our method uses the NVIDIA Isaac Gym simulation environment code (See Fig. 3) adapted from [44], [45], specifically for quadruped forward locomotion tasks, which gives critical robot state and action attributes (e.g., base position/velocity and joint positions/velocities), forming the foundational inputs to generate RFs. GPT-4o is queried to generate 8 Python-based RF samples, which we experimentally found to be an adequate number to ensure diversity in the RFs while mitigating the risk of non-executable code due to potential hallucinations by the language model. These RF samples compute various reward components (e.g., tracking of Boolean feet contact states $[4 \times 1]$, joint angles $[12 \times 1]$, and joint velocities $[12 \times 1]$), which are returned in the form of a dictionary along with the total reward value, allowing them to be tracked during training. The generated RFs are filtered to ensure they are executable Python code by checking for Python traceback errors.

C. Training Step: Isaac Gym PPO Training

The executable RFs provide the learning signal for the PPO training of the quadruped simulated agent; in our case a Unitree Go1 robot. We leverage IsaacGym's parallel training capabilities, training 4000 robots simultaneously. During PPO training for each RF, the output actions are the 12 joint positions corresponding to the agent's motors. The reward signals of the individual reward components within the RF dictionary are monitored based on the quadruped's behavior and stored every 100 iteration. These component values provide the vital signal for reward evolution in the following reward generation iteration (Sec. IV-E). For instance, if the values of one reward component are significantly larger than the rest, it may be asked to be re-scaled. If the values are unchanged throughout training, it might mean that this component cannot be optimized and is discarded.

D. Evaluation Step

For each SDS iteration, the locally optimal RF is identified and selected based on task fitness, which serves as the basis for RF evolution in following iterations. However, [20] uses a distinct human-defined fixed task fitness function for each different task, posing limitations and rendering it inappropriate for our method. Defining task fitness without MoCap systems is challenging, as SDS relies on a single video demonstration that lacks pre-engineered keypoints, camera specifications, and world context necessary for tracking-based evaluation. Additionally, GPT-4o might generate functions that do not replicate essential components of the demonstrated footage. For example, given a video of a dog walking, the reward function might guide a quadruped robot to walk, but it may fail to capture specific aspects, such as gait or style, critical to the algorithm's intended imitation objective.

The SDS evaluation step is designed to automate the assessment of the sampled RFs' imitation behavior while also addressing the limitations of [20]. To do so, for every trained RF the latest checkpoint is run for 1000 inference steps and the robot's behavior is recorded, providing video footage and the corresponding foot contact pattern plots which consists of the boolean contact states of the 4 feet of the robot (Fig. 6). To minimize computational cost and GPT-induced hallucinations, the recorded footage is also gridified to perform single-prompt evaluation rather than evaluating individual frames. Despite GPT-4V's considerable

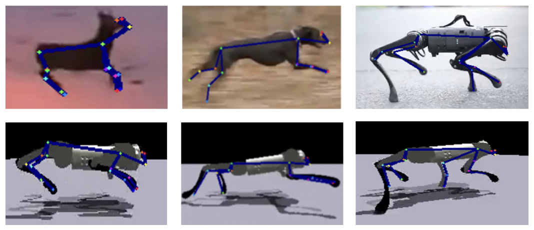


Fig. 5: ViTPose++ Pose Estimation on (top) the demonstration video and (bottom) the training sample.

capabilities in interpreting visual content, it frequently failed to accurately identify specific activities or locate the simulated robot within the video frames, e.g., GPT misclassified a quadruped as an airplane or floating bench. Hence, we incorporated ViTPose++ pose estimation technique [46] to estimate keypoints for the quadruped robot's skeleton in both the recorded inference footage and the ground truth demonstration video (Fig. 5). This approach provided a structured understanding of the quadruped's movements, facilitating a more precise evaluation of how effectively the RFs reproduced the target behaviour. The annotated frame grids and the corresponding contact sequence together form the RF evaluation prompt, used to query GPT-4o to evaluate and score the performance of each of the RFs. Each training sample is assessed on locomotion stability, gait pattern similarity, and task accuracy to identify the RF* that optimally aligns with these performance objectives. This method enhances comparative analysis and ensures the generated outputs closely align with the intended actions.

E. Reward Function Evolution

The selected locally optimal RF* from Sec. IV-D and its corresponding monitored reward components signals (Sec. IV-C) makes the evolution prompt, which serves as the basis for optimizing the RF samples generation process. GPT-4o is queried once more with this evolution prompt, SUS prompt and the gridified demonstration frames to synthesize the new evolved 8 RF samples. These evolved RFs are subsequently employed in the next SDS iteration of training, perpetuating a continuous cycle of refinement and improvement in the quadruped's behavior within the NVIDIA Isaac Gym environment.

V. Results

We verify our method using 4 videos of different quadrupeds performing different skills. The specific skills are *trot*: synchronous movement of diagonal limbs, *bound*: synchronous movement of front limbs, *pace*: synchronous movement of adjacent limbs, and *hop*: synchronous movement of all limbs. These locomotion skills are selected due to their utility in locomotion, while their proximity in behavior allows for the evaluation of SDS's sensitivity to detail apprehension. We implement SDS on the Unitree Go1's NVIDIA Jetson Xavier NX on-board computer, using a Docker-based architecture similar to [44]. The learned PPO policy for each skill runs at a 0.02s control step, with real-time policy switching enabled via the joystick. The Sim-to-Real transfer exhibits zero-shot capability, achieving seamless real-world behavior without further randomization or adaptation.

A. Skill Learning Evaluation

To validate the effectiveness of SDS in mimicking the target skill, we focus on video-to-video comparison, contact sequence analysis and stability evaluation of the learned policy. The achieved imitation is illustrated in Fig. 1, with video footage available at our Website https://rpl-cs-ucl.github.io/SDSweb/.

1) Frame-by-Frame Imitation Evaluation: Dynamic Time Warping (DTW) analysis is implemented to perform a frame-by-frame comparison between the demonstration and trained footage, due to its capabilities of handling temporal misalignment by dynamically adjusting the time windows, thereby minimizing cumulative distance. Keypoint sequences are extracted from both videos via a pose estimation model [46], followed by spatial alignment using Iterative Closest Point (ICP) analysis to mitigate spatial discrepancies between sequences. The aligned keypoints are then normalized based on their centroid and fixed scale factors, ensuring consistent scaling and positioning across frames. We perform DTW analysis for all skills, with the distance metric summarized in Table I. The DTW analysis reveals a very strong similarity between the training and demonstration footage, with higher discrepancy observed in the bounding skill. This deviation

Trot	Bound	Hop	Pace
1.28	2.85	1.47	1.92

TABLE I: DTW frame-by-frame comparison distance in the scale of e^{-6}.

correlates with tracking inaccuracies during fast movements, where motion blur in the hind legs complicates precise keypoint identification.

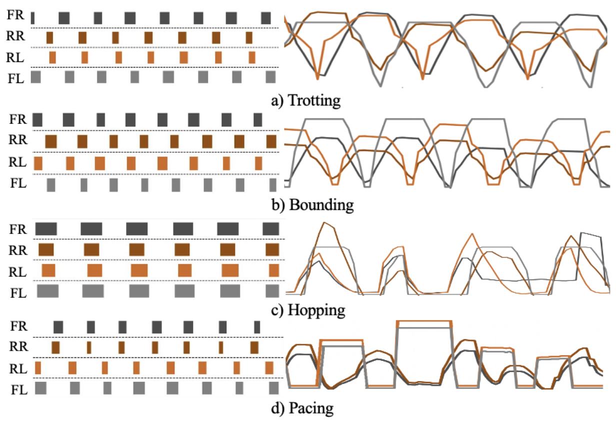


Fig. 6: Gait Evaluation: (left) Contact Sequences from simulated robot, (right) Force Sensor readings from real robot.

2) Gait Imitation Evaluation: We evaluate SDS gait imitation capabilities both in simulation and the real robot, by assessing the learned gait patterns, in the form of contact sequence and force sensor reading plots(see Fig. 6). The simulated contact sequences are obtained as boolean contact state values for all legs of the agent. For the real robot, we record force sensor readings from all legs and apply smoothing using a moving average window for visualization purposes. A clear distinction can be observed between skills, with both types of plots following the expected skill outcome, as described at the beginning of the section.

3) Locomotion Stability Evaluation: We conducted 10 experiments per skill, to assess the locomotion stability of the policies, evaluating the robot's ability to run for 1 minute in each trial. We measured the number of robot resets per run, occurring when the robot's base or knees touched the ground. All simulated experiments yield 0 resets, demonstrating reliably stable locomotion behavior. We further evaluate locomotion stability, by recording and analyzing the robot's base height fluctuations during each 1 min run (Fig. 7). Overall, all skills demonstrate a high degree of stability, characterized by periodic oscillations of low variance. Pacing and trotting exhibit minimal fluctuations in base height, indicative of steady dynamics. Bounding and hopping display denser oscillatory patterns and slightly higher variance, attributed to these gaits' faster and more dynamic nature; however, they still maintain high stability, highlighting SDS's robust control output.

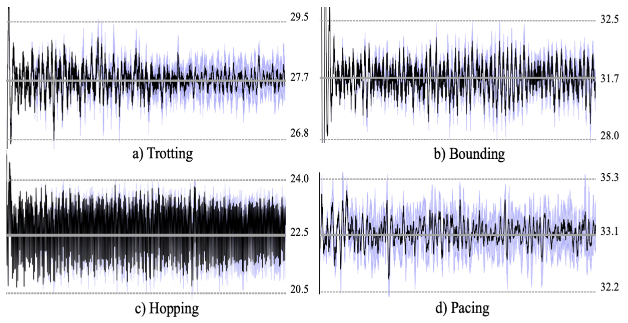


Fig. 7: Stability Evaluation: Robot base height (cm) over a consecutive 1-minute run.

B. SDS compared with SOTA methods

Our proposed SDS framework exhibits similarities on scope to Eureka [20], RoboCLIP [34], and SLoMo [30], as detailed in Sec. II. We run all three methods with our demonstration videos and can conclude the following: SDS improves upon Eureka by removing the need for manual task fitness construction and enhancing skill interpretability through visual input. Compared to RoboCLIP, SDS provides better quadrupedal locomotion performance due to the strategic use of the highly generalizable GPT-4o VLM. Despite training each skill for 4 days —3 days longer than SDS—, RoboCLIP failed to replicate the target demonstrations. SDS also demonstrates faster training than SLoMo, which takes 8 hours to process a 1-minute demonstration using 8 NVIDIA GeForce RTX 3080 GPUs. Moreover, while SLoMo requires continuous connection to a workstation for trajectory optimization and online MPC, SDS runs fully on the robot's onboard GPU, eliminating reliance on external resources.

VI. CONCLUSION

We present SDS, a new pipeline for quadrupedal skill learning from a single demo video. Using GPT-4, we automatically generate reward functions (RFs) to train PPO policies in IsaacGym, achieving effective quadruped locomotion. The key innovation lies in our prompting method and autonomous RF evaluation and evolution, enabling precise capture of skill details. SDS was validated through real-world experiments, where skill imitation was rigorously assessed using Dynamic Time Warping (DTW), achieving values on the order of e^{-6}, and gait similarity was verified through contact sequences and force sensor data, demonstrating high fidelity to the demonstration videos. Locomotion stability was further confirmed across all skills, with no reset events, and by tracking low variance in periodic base height fluctuations over one-minute intervals. SDS outperforms SOTA methods by eliminating the need for manual task-specific fitness construction, significantly reducing data requirements, and enhancing training efficiency. Future research will focus on combining skills into multi-skill policies through hierarchical learning to enable quadrupeds to learn a wider range of behaviors within one controller and extend SDS to different morphologies, such as humanoid robots.

References

[1] R. Bommasani and et al, "On the Opportunities and Risks of Foundation Models," 2022.

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

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

[4] Anthropic, "Claude 3 haiku: Our fastest model yet," March 2024, accessed: 2024-08-01.

[5] 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.

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

[7] A. Singh and et al, "FLAVA: A Foundational Language And Vision Alignment Model," 2022.

[8] X. B. Peng and et al, "Learning agile robotic locomotion skills by imitating animals," in *Robotics: Science and Systems*, 07 2020.

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

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

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: The MIT Press, 2018.

[12] D. Hadfield-Menell and et al, "Inverse Reward Design," 2020.

[13] 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*, vol. 37, no. 5, 2023, pp. 5920–5929.

[14] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[15] J. Lee, J. Hwangbo, and et al, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, Oct. 2020.

[16] D. Kanoulas, N. G. Tsagarakis, and M. Vona, "Curved patch mapping and tracking for irregular terrain modeling: Application to bipedal robot foot placement," *Robotics and Autonomous Systems*, 2019.

[17] J. Liu, S. Lyu, D. Hadjivelichkov, V. Modugno, and D. Kanoulas, "Vit-a*: Legged robot path planning using vision transformer a*," in *IEEE-RAS International Conference on Humanoids Robots (Humanoids)*, 2023.

[18] 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)*. IEEE, 2024, pp. 9264–9270.

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

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

[21] L. X. Shi and et al, "Waypoint-based imitation learning for robotic manipulation," in *Proceedings of The 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 229. PMLR, 2023, pp. 2195–2209.

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

[23] 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.

[24] 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.

[25] B. Chen and J. Su, "Addressing reward engineering for deep reinforcement learning on multi-stage task," in *Neural Information Processing*, T. Gedeon, K. W. Wong, and M. Lee, Eds. Springer International Publishing, 2019, pp. 309–317.

[26] 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.

[27] 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.

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

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

[30] Zhang and et al, "Slomo: A general system for legged robot motion imitation from casual videos," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7154–7161, 2023.

[31] X. B. Peng and et al, "Amp: Adversarial motion priors for stylized physics-based character control," in *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4. ACM, 2021, pp. 1–14.

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

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

[34] 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.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms." *CoRR*, vol. abs/1707.06347, 2017.

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

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

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

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

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

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

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

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

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

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

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