CAJun: Continuous Adaptive Jumping using a Learned Centroidal Controller

Anonymous Author(s)
Affiliation
Address
email

Abstract: We present CAJun, a novel hierarchical learning and control framework that enables legged robots to jump continuously with adaptive jumping distances. CAJun consists of a high-level centroidal policy and a low-level leg controller. In particular, we use reinforcement learning (RL) to train the centroidal policy, which specifies the gait timing, base velocity, and swing foot position for the leg controller. The leg controller optimizes motor commands for the swing and stance legs according to the gait timing to track the swing foot target and base velocity commands. Additionally, we reformulate the stance leg optimizer in the leg controller to speed up policy training by an order of magnitude. Our system combines the versatility of learning with the robustness of optimal control. We show that after 20 minutes of training on a single GPU, CAJun can achieve continuous, long jumps with adaptive distances on a Go1 robot with small sim-to-real gaps. Moreover, the robot can jump across gaps with a maximum width of 70cm, which is over 40% wider than existing methods[1].

Keywords: Jumping, Legged Locomotion, Reinforcement Learning

1 Introduction

Legged robots possess a unique capability to navigate some of the earth's most challenging terrains. By strategically adjusting their foot placement and base pose, legged robots can negotiate steep slopes [1, 2, 3], traverse uneven surfaces [4, 5], and crawl through tight spaces [6]. However, for terrains with scarce contact choices, such as gaps or stepping stones, the capability of legged robots remains somewhat limited. This limitation primarily stems from the fact that most legged robots rely heavily on walking gaits with continuous foot contacts. As such, options for foot placement are confined to within one body length from the robot's current location. Jumping offers a compelling solution to this problem. By enabling "air phases", a jumping robot can traverse through long distances without terrain contacts. Such a capability could markedly enhance a legged robot's versatility when dealing with challenging terrains. In addition, a robot capable of *continuous*, *adaptive* and *long-distance* jumps could further boost its speed and efficiency during terrain traversal.

Compared with standard walking, jumping is a significantly more challenging control task for both optimization-based [7, 8, 9, 10, 11, 6, 12, 13, 14] and learning-based controllers [15, 5, 16, 17, 18]. Optimization-based controllers, despite proving robust in challenging terrains, face computational limitations that prevent them from planning for long jumping trajectories in real time. Typically, these controllers circumvent this issue by first solving an intricate trajectory optimization problem offline, then utilizing simplified model predictive control (MPC) to track this predetermined *fixed* trajectory online. Consequently, existing works tend to be restricted to non-adaptive, single jumps [6, 12, 13, 14]. On the other hand, RL controllers have the potential to learn more adaptive and versatile locomotion skills, but they require substantial effort in reward design and sim-to-real transfer [19, 15, 20, 4],

[1] Videos in https://sites.google.com/view/continuous-adaptive-jumping

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

particularly for dynamic and underactuated tasks such as jumping. Therefore, achieving continuous jumping over long distances can be a significant challenge for existing methods.

In this paper, we present CAJun (Continuous Adaptive Jumping with a Learned Centroidal Policy), which achieves continuous long-distance jumpings with adaptive distances on the real robot. Our framework seamlessly combines optimization-based control and RL in a hierarchical manner. Specifically, a high-level RL-based *centroidal policy* specifies the desired gait, target base veloctiy, and swing foot positions to the *leg controller*, and a low-level *leg controller* solves the optimal motor commands given the centroidal policy's action. Our framework effectively integrates the benefits of both control and learning. First, the RL-based *centroidal policy* is able to learn versatile, adaptive jumping behaviors without heavy computational burden. Second, the low-level quadratic-programming-based (QP) *leg controller* optimizes torque commands at high frequency (500Hz), which ensures reactive feedback to environmental perturbations and significantly reduces the sim-to-real gap. Finally, to resolve the common training speed bottleneck in hierarchical methods [21, 22, 23], we reformulated the QP problem in the *leg controller* to a least-squares problem with clipping so that the entire stack is 10 times faster and can be executed in massive parallel [16].

Within 20 mins of training in simulation, we deploy CAJun directly to a Unitree Go1 robot [24]. Without any fine-tuning, CAJun achieves continuous, long-distance jumping, and adapts its jumping distance based on user command. Moreover, using the alternating contact pattern in a bounding gait, the robot is capable of crossing a gap of 70cm, which is at least 40% larger than existing methods (Fig. 4 and Table 1). To the best of our knowledge, CAJun is the first framework that achieves continuous, adaptive jumping with such gap-crossing capability on a commercially available quadrupedal robot. We further conduct ablation studies to validate essential design choices. In summary, our contribution with CAJun are the following:

- We present CAJun, a hierarchical learning and control framework for continuous, adaptive, long-distance jumpings on legged robots.
- We demonstrate that jumping policies trained with CAJun can be directly transferred to the real world with a gap-crossing capability of 70cm.
- We show that CAJun can be trained efficiently in less than 20 minutes using a single GPU.

2 Related Works

Optimization-based Control for Jumping Using optimization-based controllers, researchers have achieved a large variety of jumping behaviors, from continuous pronking and bounding [7, 8, 9, 10, 11] to large single-step jumps [6, 12, 13, 14]. By optimizing for control inputs at a high frequency, these controllers can execute robust motions even under severe perturbations [8, 9]. However, due to the high computation cost, they cannot plan ahead for a long horizon during online execution. Therefore, they primarily focus on high-frequency jumps with a short CoM displacement per jump [9, 10, 11]. One way to overcome this computation limit is to pre-compute a reference trajectory offline using trajectory optimization (TO) [6, 12, 13, 14], which can greatly extend the height [12] and distance [13] of each jump. However, it can be challenging to generalize beyond the reference trajectories towards continuous, adaptive jumping [25, 26, 27]. Notably, using a multi-level planner, Park et al. [26] achieved continuous bounding with fixed gait and adaptive height to jump over hurdles. Compared to these approaches, our framework adopts a more general formulation, where the policy can adjust the gait timing, base pose, and swing foot position simultaneously.

Learning-based Control for Jumping In recent years, learning-based controllers have significantly improved the capability of legged robots, from rapid running [28] to traversing over challenging terrains [5]. While standard walking gaits can be learned from scratch using reinforcement learning (RL), more dynamic behaviors such as jumping usually require additional setup in the learning process, such as motion imitation [19, 18, 17], curriculum learning [16] and multi-stage training [3, 29]. Another challenge for learning-based controllers is sim-to-real, especially for dynamic

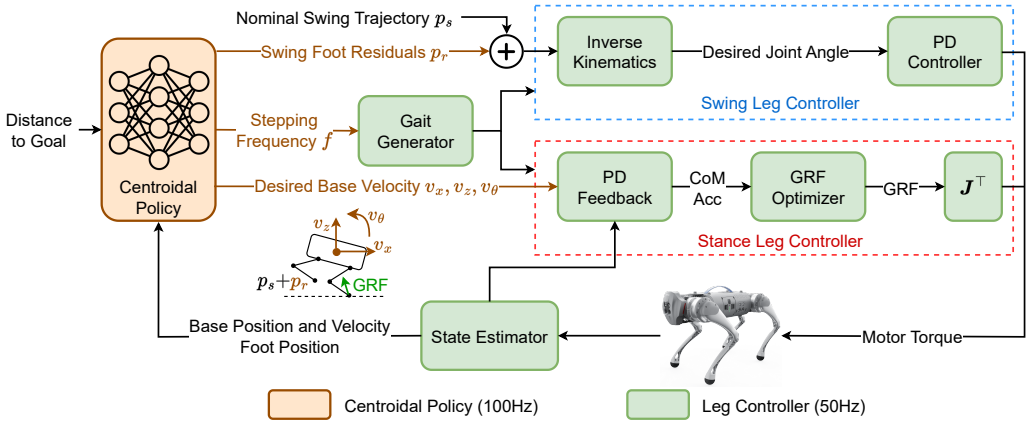

Figure 1: Overview of the hierarchical framework of CAJun.

underactuated behaviors like jumping [30]. To overcome the sim-to-real gap, researchers have developed a suite of tools such as domain randomization [15], system identification [31] and motor adaptation [20]. Recently, Smith et al. [19] used motion imitation and transfer learning to jump over a gap of 20cm (0.4 body length) on a Unitree A1 robot, and Caluwaerts et al. [3] used multi-stage training with policy synthesis to jump over a gap of 50cm (1 body length) on a custom-built quadrupedal robot. Compared to these works, CAJun's hierarchical setup can jump over *wider* gaps (70cm / 1.4 body length) *continuously*, and can adapt its landing position based on user command.

Hierarchical RL for Legged Robots Recently, there has been increasing interest in combining RL with optimization-based control for legged robots [22, 32, 21, 23, 33, 34]. These frameworks typically follow a hierarchical structure, where a high-level RL-trained policy outputs intermediate commands to a low-level leg controller. The RL policy can give several forms of instructions to the low-level controller, such as gait timing [22, 32], CoM trajectory [21, 34, 30, 33] and foot landing positions [23, 35, 36, 37, 38]. Our approach uses a similar hierarchical setup but adopts a general action space design where the policy specifies the gait, CoM velocity and swing foot locations *simultaneously*. One bottleneck of the hierarchical approaches is the slow training time because every environment rollout involves solving the optimization problem in the low-level controller. We overcome this bottleneck by relaxing the constraints in foot force optimization, so that foot force can be solved efficiently in closed form. Compared to existing frameworks which can take hours or even days to train, CAJun can be trained in 20 minutes using GPU-accelerated simulation [16].

3 Overview of CAJun

In order to learn continuous, long-distance, and adaptive jumping behaviors, we design CAJun as a hierarchical framework consisting of a high-level centroidal policy and a low-level leg controller (Fig. 1). To specify a jump, The *centroidal policy* outputs three key actions to the low-level controller, namely, the stepping frequency, the swing foot residual, and the desired base velocity. The modules in the *leg controller* then convert these actions into motor commands. Similar to previous works [8, 9, 32, 22, 21], the *leg controller* adopts separate control strategy for swing and stance legs, where the desired contact state of each leg is determined by the *gait generator*. We design the gait generator to follow a pre-determined contact sequence with timings adjustable by the high-level centroidal policy. For swing legs, we first find its desired position based on a heuristically-determined reference trajectory and learned residuals, and converts that to joint position commands using inverse kinematics. For stance legs, we first determine the desired base acceleration from the policy commands, and then solves an optimization problem to find the corresponding Ground Reaction Forces (GRFs) to reach this acceleration. We run the low-level controller at 500Hz for fast, reactive torque control, and the high-level controller at 100Hz to ensure stable policy training.

4 Low-level Leg Controller

Similar to prior works [22, 21, 8], the low-level controller of CAJun adopts separate control strategies for swing and stance legs, and uses a gait generator to track the desired contact state of each leg.

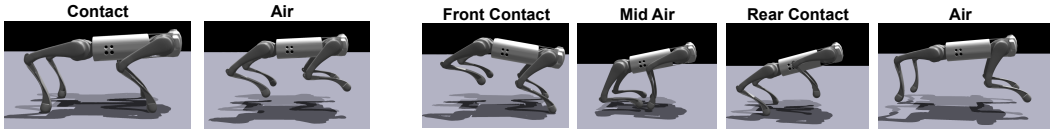

Figure 2: The contact sequence and default timing of the pronking (**left**) and bounding (**right**) gait.

Additionally, we carefully design the interface between the centroidal policy and components in the leg controller to maintain control robustness and policy expressiveness. Moreover, we relaxed the GRF optimization problem in stance leg controller to significantly speed up training.

4.1 Phase-based Gait Generator

The gait generator determines the desired contact state of each leg (swing or stance) based on a pre-defined contact sequence and the timing information from the centroidal policy. To capture the cyclic nature of locomotion, we adopt a phase-based gait representation, similar to prior works [22, 39]. The gait is modulated by a phase variable ϕ, which increases monotonically from 0 to 2π in each locomotion cycle, and wraps back to 0 to start the next cycle. The propagation of ϕ is controlled by the *stepping frequency* f, which is commanded by the centroidal policy:

$$\phi_{t+1} = \phi_t + 2\pi f \Delta t \tag{1}$$

where Δt is the control timestep. The mapping from ϕ to the desired contact state is pre-defined. We adopt two types of jumping gaits in this work, namely, *bounding* and *pronking*, where bounding alternates between the front and rear leg contacts, and pronking lands and lifts all legs at the same time (Fig. 2). Note that while the *sequence* of contacts is fixed, the centroidal policy can flexibly adjust the *timing* of contacts to based on the state of the robot.

4.2 Stance Leg Control

The stance leg controller computes the desired joint torque given the velocity command from the centroidal policy. Since jumping is mostly restricted to the sagittal plane, the policy specifies the velocity in the forward and upward axis (v_x, v_z), as well as the rotational velocity v_θ, and the velocity for the 3 remaining DoF is set to 0. We compute the desired torque following a 3-step procedure. First, we compute the desired CoM acceleration $\ddot{\boldsymbol{q}}^{\text{ref}} \in \mathbb{R}^6$ using a PD controller (Appendix. A.2). Next, we optimize for the GRF $\boldsymbol{f} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3, \boldsymbol{f}_4] \in \mathbb{R}^{12}$ to track this desired acceleration, where $\boldsymbol{f}_i$ is the foot force vector of leg i. Lastly, we compute the motor torque command using $\boldsymbol{\tau} = \boldsymbol{J}^\top \boldsymbol{f}$, where $\boldsymbol{J}$ is the foot Jacobian. Since the GRF optimization is typically the computation bottleneck for optimization-based controllers [21, 22, 23], we re-design this optimization procedure to significantly speed up the training process.

QP-based GRF Optimization To optimize for GRF, prior works typically solve the following quadratic program (QP):

$$\min_{\boldsymbol{f}} \ \|\ddot{\boldsymbol{q}} - \ddot{\boldsymbol{q}}^{\text{ref}}\|_{\mathbf{U}} + \|\boldsymbol{f}\|_{\mathbf{V}} \tag{2}$$

$$\text{subject to: } \ddot{\boldsymbol{q}} = \mathbf{A}\boldsymbol{f} + \boldsymbol{g} \tag{3}$$

$$f_{i,z} = 0 \qquad \text{if } i \text{ is a swing leg} \tag{4}$$

$$f_{\text{min}} \leq f_{i,z} \leq f_{\text{max}} \qquad \text{if } i \text{ is a stance leg} \tag{5}$$

$$-\mu f_{i,z} \leq f_{i,x} \leq \mu f_{i,z}, \quad -\mu f_{i,z} \leq f_{i,y} \leq \mu f_{i,z} \qquad i = 1, \ldots, 4 \tag{6}$$

Eq. (3) represents the centroidal dynamics model [8], where $\mathbf{A}$ is the generalized time-variant inverse inertia matrix, and $\boldsymbol{g}$ is the gravity vector (see Appendix. A.2 for details). Eq. (4), (5) specifies the contact schedule, as computed by the gait generator. Eq. (6) specifies the approximated friction cone constraints, where μ is the friction coefficient. $\mathbf{U}, \mathbf{V} \succ 0$ are positive definite weight matrices.

Unconstrained GRF Optimization with Clipping QP-based GRF optimization would require an iterative procedure (e.g., active set method and interior point method), which can be computationally expensive and difficult to scale up in parallel in GPU. Instead of using the QP formulation, CAJun relaxes this optimization problem by solving the unconstrained GRF first and clipping the resulting GRF to be within the friction cone. Since Eq. (3) is linear in $\boldsymbol{f}$, if we ignore the constraints in Eq. (6) and (5) and eliminate the variables for non-contact legs, the optimal $\boldsymbol{f}$ can be solved *in closed-form*:

$$\widehat{\boldsymbol{f}} = (\mathbf{A}^\top \mathbf{U} \mathbf{A} + \mathbf{V})^{-1} \mathbf{A}^\top \mathbf{U} (\ddot{\boldsymbol{q}}^{\text{ref}} - \boldsymbol{g}) \tag{7}$$

Next, we clip the solved ground reaction forces to lie within the friction cone:

$$f_{i,z} = \text{clip}(\widehat{f}_{i,z}, f_{\text{min}}, f_{\text{max}}), \quad (f_{i,x}, f_{i,y}) = (\widehat{f}_{i,x}, \widehat{f}_{i,y}) \cdot \min(1, \mu f_{i,z} / \sqrt{\widehat{f}_{i,x}^2 + \widehat{f}_{i,y}^2}) \tag{8}$$

Note that our unconstrained formulation not only reduces computational complexity, but also makes the solving procedure highly parallelizable. Therefore, when paired with GPU-accelerated simulator like Isaac Gym [16], CAJun can be trained efficiently in massive parallel, which significantly reduced the turn-around time. Additionally, while our unconstrained formulation may yield sub-optimal solutions when the least-squares solution (Eq. (7)) finds a GRF outside the friction cone, the high-level *centroidal policy* would observe this sub-optimality during training, and thereby compensating for it by adjusting the desired CoM velocity commands. In practice, we find the policies trained using the constrained and unconstrained optimization to perform similarly (see Sec. 6.6 for details).

4.3 Swing Leg Control

We use position control for swing legs, where the desired position is the sum of a heuristically constructed reference trajectory [8, 40] and a residual output from the centroidal policy. Similar to prior works [22, 21], we generate the reference trajectory by interpolating between key points in the swing phase (see Appendix. A.3 for details). On top of the heuristic trajectory ($\boldsymbol{p}_s$ in Fig. 1), the centroidal policy adjusts the swing foot trajectory for higher foot clearance and optimal foot placement by outputting a residual in foot position ($\boldsymbol{p}_r$ in Fig. 1). Once the foot position is determined, we convert it to desired motor angles using inverse kinematics and execute it using joint PD commands.

5 Learning a Centroidal Policy for Jumping

The RL problem is represented as a Markov Decision Process (MDP), which includes the state space $\mathcal{S}$, action space $\mathcal{A}$, transition probability $p(s_{t+1}|s_t, a_t)$, reward function $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, and initial state distribution $p_0(s_0)$. We aim to learn a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ that maximizes the expected cumulative reward over an episode of length T, which is defined as $J(\pi) = \mathbb{E}_{s_0 \sim p_0(\cdot), s_{t+1} \sim p(\cdot|s_t, \pi(s_t))} \sum_{t=0}^{T} r(s_t, a_t)$.

Environment Overview For maximum expressiveness, we design the environment such that the centroidal policy directly specifies the contact schedule, base velocity and swing foot position for the low-level controller. To focus on continuous jumps, we design each episode to contain exactly 10 jumping cycles, where termination is determined by the gait generator (Section. 4.1). Additionally, we normalize the reward so that total reward within each jumping cycle is agnostic to its duration. In order to learn distance-adaptive jumping, we sample different jumping distances uniformly in [0.3m, 1m] before each jump, and compute the desired landing position, which is included in the state space.

State and Action Space We design the state space to include the robot's proprioceptive state, as well as related information about the current jump. The proprioceptive information includes the current position and velocity of the robot base, as well as the foot positions in the base frame. The task information includes the current phase of the jump ϕ (Sec. 4.1) and the location of the target landing position in egocentric frame. The action space includes the desired stepping frequency f, the desired base velocity in sagittal plane v_x, v_z, v_θ, as well as the desired swing foot residuals, which are specified to different modules in the low-level controller.

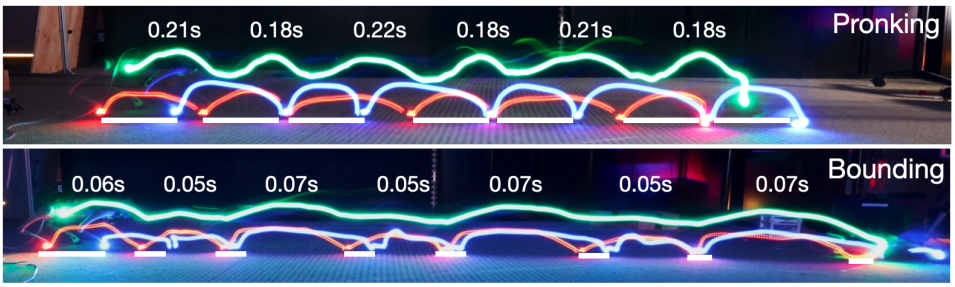

Figure 3: Long-exposure photos visualizing base (green), front foot (blue) and rear foot (red) trajectories of the robot when jumping with alternating distance commands. White lines show the foot positions during each landing (contact phase for pronking, mid-air phase for bounding). Time shows the duration of "air phase" (Fig. 2) in each jump when all legs are in the air.

Reward Function We design a reward function with 9 terms. At a high level, the reward function ensures that the robot maintains an upright pose, follows the desired contact schedule, and lands close to goal. See Appendix. B.1 for the detailed weights and definitions.

Early Termination To speed up training and avoid unnecessary exploration in sub-optimal states, we terminate an episode early if the robot's base height is less than 15cm, or the base orientation deviates significantly from the upright pose.

Policy Representation and Training We represent policy and value functions using separate neural networks. Each network includes 3 hidden layers of $[512, 256, 128]$ units respectively with ELU activations [41]. We train our policy using Proximal Policy Optimization (PPO) [42]. Please see Appendix. B.2 for the detailed configuration.

6 Results and Analysis

We design experiments to validate that CAJun can learn continuous and adaptive jumping controllers. In particular, we aim to answer the following questions:

1. Can CAJun enable the robot to learn continuous jumping with adaptive jumping distances?
2. What is the widest gap that the robot can jump over using CAJun?
3. How robust is the learned jumping controller against external perturbations?
4. What is the advantage of the hierarchical design of CAJun, and what are important design choices?

6.1 Experiment Setup

We use the Go1 quadrupedal robot from Unitree [24] as our experiment platform, and build the simulation in IsaacGym [16, 43] for fast training. To match the GPU-accelerated, massively-parallel simulation environment, we implement the entire control stack, including the centroidal policy and leg controller, in a vectorized form in PyTorch [44]. We adopt the PPO implementation from `rsl_rl` [16], which is specifically optimized for GPU-accelerated environments. We train CAJun on a standard desktop with an Nvidia RTX 2080Ti GPU, which takes less than 20 minutes to complete.

6.2 Continuous and Adaptive Jumping

To verify that CAJun can learn continuous, dynamic jumping with adaptive jumping distances on the real robot, we deploy the trained *pronking* and *bounding* controllers to the real robot. For each gait, we run it continuously for at least 6 jumps, where the desired jumping distance alternates between 0.3 and 1 meter. We put LEDs on the base and feet of the robot and capture the robot's trajectory using long-exposure photography (Fig. 3).

We find that both the pronking and the bounding controller can be deployed successfully to the real robot, and achieve continuous jumping with long jumping distances. Both the base and the

Figure 4: Using the bounding gait, the robot can jump over a 60cm-wide yoga mat without making foot contact.

Method	Jumping Style	Widest Gap Crossed
TWiRL [19]	Single	0.2m
Barkour [3]	Single	0.5m
Margolis et al. [30]	Continuous	0.26m
CAJun (ours)	Continuous w/ Adaptive Jumping Distance	**0.7m**

Table 1: Comparison of gap-crossing capability on controllers deployed to similar-sized robots.

foot trajectories exhibit clear periodicity, which demonstrates the long-term stability of the jumping controller. Moreover, the policy responds to jumping distance commands well, and results in alternating patterns of further and closer jumps. A closer look at the duration of each air phase shows that in both the bounding and pronking gait, the centroidal policy reduces the air time by approximately 20% when switching from longer to shorter jumps. This is achieved by the stepping frequency output (Section. 4.1) of the centroidal policy. As demonstrated in previous works [22, 45], such gait adjustments can potentially save energy and extend the robot's operation time.

6.3 Jumping over Wide Gaps

While both the pronking and bounding gait can jump with at least 70cm of base movement in each step, we find that the bounding gait offers a unique advantage in traversing through gaps. As seen in the foot trajectories in Fig. 3, the alternating contact pattern in bounding enables the front and rear of the robot to land closely in the world frame, so that the robot can utilize *the entire jumping distance* of 70cm for gaps. To further validate this, we place a yoga mat with a width of 60cm in the course of the robot, and find that the robot can jump over it with additional buffer space before and after the jump (Fig. 4). To the best of our knowledge, CAJun is the first framework that achieves continuous jumping with such gap-crossing capability on a commercially-available quadrupedal robot (Table. 1).

6.4 Validation on Robustness

We design two experiments to further validate the robustness of CAJun. In the first experiment, we add a leash to the back of the robot and actively pulled the leash during jumping (Fig. 5). While both the pronking and bounding gait experienced a significant drop in forward velocity during the pull, they recovered from the pull and regained momentum for subsequent jumps. In the second experiment, we test the robot outdoors, where the robot needs to jump from asphalt to grass (Fig. 6). The uneven and slippery surface of the grass perturbed the robot and broke the periodic pattern in pitch angles. However, both policies recovered from the initial perturbation, and resume stable, periodic jumps after around 2 jumping cycles. The robustness of CAJun can be likely attributed to the high control frequency of the low-level leg controller, which enables the robot to react swiftly to unexpected perturbations, and the online adjustment of the learned centroidal policy.

6.5 Comparison with End-to-End RL

To demonstrate the effectiveness of CAJun's hierarchical setup, we compare it to an end-to-end RL baseline, where the policy directly outputs motor position commands. Please refer to Appendix. B.3 for the setup details. In both simulation and the real world, we run each policy for 6 jumps with a desired distance of 1 meter per jump, and report the total CoM displacement in Table. 2. In the real world, CAJun outperforms the e2e baseline for both gaits (25% further in bounding, 185% in pronking). Additionally, note that CAJun faces a significantly smaller sim-to-real gap, where the total jumping distance drops by no more than 5% when transferred from sim-to-real. In comparison, the performance drop for e2e baseline can be as large as 60%. While additional sim-to-real efforts

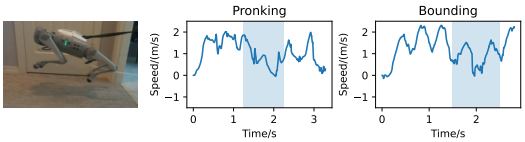

Figure 5: Forward velocity of the robot jumping under leash pulling (shaded area shows active pulling).

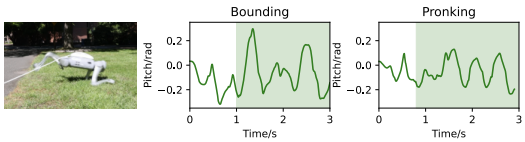

Figure 6: Pitch angle of the robot jumping from asphalt to grass (shaded area indicates grass)

	Pronking		**Bounding**	
	Sim	Real	Sim	Real
E2E	4.17±0.01	1.67±0.18	**4.61±0.03**	3.47±0.15
CAJun (ours)	**4.98±0.02**	**4.76±0.11**	4.27±0.05	**4.34± 0.17**

Table 2: Total distance after 6 jumps achieved by end-to-end RL and CAJun.

such as domain randomization [15], system identification [31] or teacher-student training [20] might reduce this gap for e2e methods, the hierarchical framework of CAJun offers a simple and efficient alternative that can be deployed zero-shot to the real world.

6.6 Ablation Study

We design a set of ablation studies to validate the design choices of CAJun. We summarize the results here. Please refer to Appendix. B.4 for setup details and additional results.

No Gait Modulation The stepping frequency from the centroidal policy is essential for the stability of the robot. In *no-gait*, we disable the stepping frequency output and adopt a fixed stepping frequency of 1.66Hz for both the pronking and bounding gait, which is the average stepping frequency output from CAJun. While the baseline can achieve a similar reward, the learning process is noisy with frequent failures. Since the heuristically-designed gait might not match the capability of the robot, it is important for the policy to adjust the gait timing to stabilize each jump.

No Swing Leg Residual The swing residuals play a critical role in achieving long-distance jumps. To validate that, we design a baseline, *no-swing*, where we disable the swing residuals so that swing legs completely follow the heuristically-designed trajectory from the swing controller. We find that the baseline policy cannot jump as far as CAJun, and achieves a lower reward for both gaits.

CAJun-QP The clipped QP in GRF optimization significantly reduced training time without noticeable performance drops. To validate this design choice, we compare the training time and policy performance of CAJun with a variant, CAJun-QP, where we solve for GRFs using the complete QP setup, where the approximated friction cone is imposed as constraints. We adopt the QP-solver from `qpth` [46], an efficient interior-point-method-based solver that supports GPU acceleration. For both the pronking and bounding gait, we find that CAJun achieves a similar reward compared to CAJun-QP. However, because CAJun-QP needs to iteratively optimize GRF at every control step, its training time is almost 10 times longer, which is consistent with prior observations [21].

7 Limitations and Future Work

In this work, we present CAJun, a hierarchical learning framework for legged robots that consists of a high-level centroidal policy and a low-level leg controller. CAJun can be trained efficiently using GPU-accelerated simulation and can achieve continuous jumps with adaptive jumping distances of up to 70cm. One limitation of CAJun is that, while it can adapt to changes in jumping distances, it can not land accurately at the desired location yet. This inaccuracy might be due to a number of factors such as unmodeled dynamics and state estimation drifts. Another limitation of CAJun is that it does not make use of perception, and only adjusts its jumping distances based on ad-hoc user commands. In future work, we plan to extend CAJun to incorporate perception and achieve more accurate jumps, so that the robot can demonstrate extended agility and autonomy in challenging terrains.

References

[1] C. Gehring, C. D. Bellicoso, S. Coros, M. Bloesch, P. Fankhauser, M. Hutter, and R. Siegwart. Dynamic trotting on slopes for quadrupedal robots. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5129–5135. IEEE, 2015.

[2] H. Kolvenbach, P. Arm, E. Hampp, A. Dietsche, V. Bickel, B. Sun, C. Meyer, and M. Hutter. Traversing steep and granular martian analog slopes with a dynamic quadrupedal robot. *arXiv preprint arXiv:2106.01974*, 2021.

[3] K. Caluwaerts, A. Iscen, J. C. Kew, W. Yu, T. Zhang, D. Freeman, K.-H. Lee, L. Lee, S. Saliceti, V. Zhuang, et al. Barkour: Benchmarking animal-level agility with quadruped robots. *arXiv preprint arXiv:2305.14654*, 2023.

[4] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[5] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on Robot Learning*, pages 403–415. PMLR, 2023.

[6] S. Gilroy, D. Lau, L. Yang, E. Izaguirre, K. Biermayer, A. Xiao, M. Sun, A. Agrawal, J. Zeng, Z. Li, et al. Autonomous navigation for quadrupedal robots with optimized jumping through constrained obstacles. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 2132–2139. IEEE, 2021.

[7] C. D. Bellicoso, F. Jenelten, P. Fankhauser, C. Gehring, J. Hwangbo, and M. Hutter. Dynamic locomotion and whole-body control for quadrupedal robots. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3359–3365. IEEE, 2017.

[8] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1–9. IEEE, 2018.

[9] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim. Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *arXiv preprint arXiv:1909.06586*, 2019.

[10] Y. Ding, A. Pandala, and H.-W. Park. Real-time model predictive control for versatile dynamic motions in quadrupedal robots. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8484–8490. IEEE, 2019.

[11] C. Gehring, S. Coros, M. Hutter, C. D. Bellicoso, H. Heijnen, R. Diethelm, M. Bloesch, P. Fankhauser, J. Hwangbo, M. Hoepflinger, et al. Practice makes perfect: An optimization-based approach to controlling agile motions for a quadruped robot. *IEEE Robotics & Automation Magazine*, 23(1):34–43, 2016.

[12] Q. Nguyen, M. J. Powell, B. Katz, J. Di Carlo, and S. Kim. Optimized jumping on the mit cheetah 3 robot. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7448–7454. IEEE, 2019.

[13] Z. Song, L. Yue, G. Sun, Y. Ling, H. Wei, L. Gui, and Y.-H. Liu. An optimal motion planning framework for quadruped jumping. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11366–11373. IEEE, 2022.

[14] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, 2018.

[15] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.

[16] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[17] G. B. Margolis and P. Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In *Conference on Robot Learning*, pages 22–31. PMLR, 2023.

[18] A. Klipfel, N. Sontakke, R. Liu, and S. Ha. Learning a single policy for diverse behaviors on a quadrupedal robot using scalable motion imitation. *arXiv preprint arXiv:2303.15331*, 2023.

[19] L. Smith, J. C. Kew, T. Li, L. Luu, X. B. Peng, S. Ha, J. Tan, and S. Levine. Learning and adapting agile locomotion skills by transferring experience. *arXiv preprint arXiv:2304.09834*, 2023.

[20] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.

[21] Z. Xie, X. Da, B. Babich, A. Garg, and M. v. de Panne. Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model. In *Algorithmic Foundations of Robotics XV: Proceedings of the Fifteenth Workshop on the Algorithmic Foundations of Robotics*, pages 523–539. Springer, 2022.

[22] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots. Fast and efficient locomotion via learned gait transitions. In *Conference on Robot Learning*, pages 773–783. PMLR, 2022.

[23] W. Yu, D. Jain, A. Escontrela, A. Iscen, P. Xu, E. Coumans, S. Ha, J. Tan, and T. Zhang. Visual-locomotion: Learning to walk on complex terrains with vision. In *5th Annual Conference on Robot Learning*, 2021.

[24] Unitree. Go1 Website. URL https://www.unitree.com/products/go1/.

[25] C. Nguyen, L. Bao, and Q. Nguyen. Continuous jumping for legged robots on stepping stones via trajectory optimization and model predictive control. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 93–99. IEEE, 2022.

[26] H.-W. Park, P. M. Wensing, and S. Kim. Jumping over obstacles with mit cheetah 2. *Robotics and Autonomous Systems*, 136:103703, 2021.

[27] H.-W. Park, P. M. Wensing, S. Kim, et al. Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. 2015.

[28] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid locomotion via reinforcement learning. *arXiv preprint arXiv:2205.02824*, 2022.

[29] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Robust and versatile bipedal jumping control through multi-task reinforcement learning. *arXiv preprint arXiv:2302.09450*, 2023.

[30] G. B. Margolis, T. Chen, K. Paigwar, X. Fu, D. Kim, S. bae Kim, and P. Agrawal. Learning to jump from pixels. In *Conference on Robot Learning*, pages 1025–1034. PMLR, 2022.

[31] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[32] X. Da, Z. Xie, D. Hoeller, B. Boots, A. Anandkumar, Y. Zhu, B. Babich, and A. Garg. Learning a contact-adaptive controller for robust, efficient legged locomotion. In *Conference on Robot Learning*, pages 883–894. PMLR, 2021.

[33] Y. Yang, X. Meng, W. Yu, T. Zhang, J. Tan, and B. Boots. Continuous versatile jumping using learned action residuals. *arXiv preprint arXiv:2304.08663*, 2023.

[34] G. Bellegarda and Q. Nguyen. Robust quadruped jumping via deep reinforcement learning. *arXiv preprint arXiv:2011.07089*, 2020.

[35] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis. Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control. *arXiv preprint arXiv:2012.03094*, 2020.

[36] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter. Robust rough-terrain locomotion with a quadrupedal robot. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5761–5768. IEEE, 2018.

[37] O. Villarreal, V. Barasuol, P. M. Wensing, D. G. Caldwell, and C. Semini. Mpc-based controller with terrain insight for dynamic legged locomotion. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2436–2442. IEEE, 2020.

[38] F. Jenelten, T. Miki, A. E. Vijayan, M. Bjelonic, and M. Hutter. Perceptive locomotion in rough terrain–online foothold optimization. *IEEE Robotics and Automation Letters*, 5(4):5370–5376, 2020.

[39] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke. Policies modulating trajectory generators. In *Conference on Robot Learning*, pages 916–926. PMLR, 2018.

[40] M. H. Raibert. *Legged robots that balance*. MIT press, 1986.

[41] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

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

[43] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021.

[44] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[45] Z. Fu, A. Kumar, J. Malik, and D. Pathak. Minimizing energy consumption leads to the emergence of gaits in legged robots. *arXiv preprint arXiv:2111.01674*, 2021.

[46] B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*, pages 136–145. PMLR, 2017.