Agile and Versatile Robot Locomotion via Kernel-based Residual Learning

Milo Carroll[1], Zhaocheng Liu[1], Mohammadreza Kasaei[1] and Zhibin Li[2]

***Abstract*— This work developed a kernel-based residual learning framework for quadrupedal robotic locomotion. Initially, a kernel neural network is trained with data collected from an MPC controller. Alongside a frozen kernel network, a residual controller network is trained using reinforcement learning to acquire generalized locomotion skills and robustness against external perturbations. The proposed framework successfully learns a robust quadrupedal locomotion controller with high sample efficiency and controllability, which can provide omnidirectional locomotion at continuous velocities. We validated its versatility and robustness on unseen terrains that the expert MPC controller failed to traverse. Furthermore, the learned kernel can produce a range of functional locomotion behaviors and can generalize to unseen gaits.**

I. INTRODUCTION

The versatility of legged locomotion exceeds other forms, such as wheeled locomotion, which requires continuous ground support and cannot feasibly adapt to challenging terrains [1], [2]. While quadrupedal animals access the most remote locations by exploring terrains that are never seen before [3], other forms of robots usually would fail to do so.

Traditional optimisation-based controllers perform well in challenging terrains [4], [5], [6], [7]. However, due to high computation demands, they are prone to external perturbations and large model errors [3], [8]. Recently, Deep Reinforcement Learning (DRL) methods have resulted in many robust locomotion controllers that operate at much higher frequencies enabling higher resiliency against errors and perturbations. However, RL-based controllers usually require carefully designed reward functions and excessive training data to produce an efficient controller with natural gaits [9], [10], [11], [12]. Additionally, with the disagreement between physics simulators and the real world, DRL controllers also face the sim-to-real gap when testing on a real robot [13].

Many legged animals start walking shortly after birth [14] due to pre-developed neural circuits, which are refined rapidly to acquire expert skills. Inspired by this, Residual learning (ResL) is introduced for training RL agents only to adapt a prior control behavior, quickly learning robust and natural legged locomotion [15], [16], [17], [18], [19]. ResL methods can be grouped by the approach of providing the control priors: library-based, controller-based, and learning-based methods, of which have emerged chronologically. We break these down in the following subsections.

[1] Milo Carroll, Zhaocheng Liu and Mohammadreza Kasaei are with the School of Informatics, University of Edinburgh, UK. Email: {S2173175, zc.liu, m.kasaei}@ed.ac.uk

[2] Zhibin Li is with the Department of Computer Science, University College London, UK. Email: alex.li@ucl.ac.uk

A. Residual Learning (ResL) Methods

Library-based references. These methods use pre-defined trajectory loops, which are static and can be quired to provide references [17], [18]. They have been shown to produce robust and versatile locomotion in a sample efficient manner requiring less than 10M timesteps to converge, only requiring a library consisting of a single loop [17]. Improved velocity control can be achieved with a gait library providing trajectories queried by the target velocity. Yet, this can only provide priors for discrete velocities. Thus continuous velocity control requires the agent to work against the prior rather than working with it.

Controller-based references. These methods leverage existing expert controllers to provide the priors within a ResL framework, [19] and [20], using MPC and CPG-based controllers, respectively. This is beneficial, as the expert controller provides omnidirectional locomotion priors with continuous velocity control. However, as the controllers are adaptive to the robots state, the residual agent must learn to model how these controllers respond, thus making the RL problem considerably more challenging. This is further reflected in the sample efficiency, with [19] and [20] both requiring over 100M training timesteps to converge – considerably more than the library-based methods.

Learning references. Learning has been incorporated into the reference generation process in various ways [21], [22], [23]. One approach uses a linear layer to adapt the trajectories produced by a CPG controller [21], producing more suitable trajectory priors for specific terrains. In [23], a kernel is learned using a conditional variational auto-encoder (cVAE [24]) from a motion database. The method provides the desired omnidirectional locomotion, velocity control, and versatility. Nevertheless, the sample efficiency of the method remains weak (200M). When priors are stochastic [23] or adaptive [19], [20], poor sample efficiency emerges. Thus, here is an identified gap to achieve sample efficiency similar to library-based methods and functionality comparable to controller-based methods, in terms of generating omnidirectional priors that are deterministic, non-adaptive, and provide continuous velocity control.

B. Learning Trajectory-based Controllers

Kinematic Motion Primitives (KMPs) [25] are used for developing data-driven locomotion controllers [26], but they produce static gaits and have no adaption, e.g. walking at continuous target velocities. Similar controllability problems exist in Dynamic Movement Primitives (DMPs) [27], [28], [29]. Although, a trained DMP's hyper-paramters can be

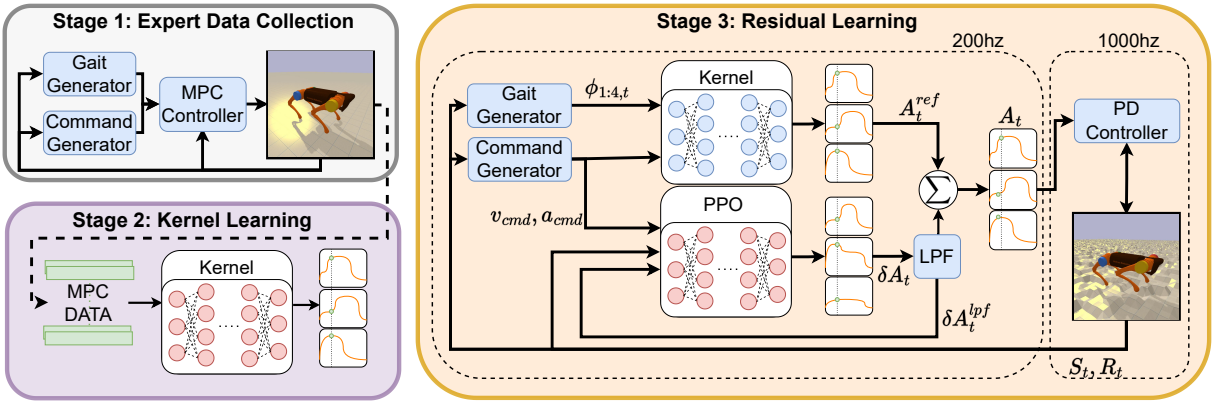

Fig. 1: Overview of the proposed multi-stage robot locomotion framework, where the red components represent trainable modules, and blue components represent fixed modules.

tuned to adjust amplitude, frequency, and offset of the trajectories, showing potential for adaptive control. FastMimic [30] exploits this, optimizing DMPs fitted to retargeted motion capture data, demonstrating rapid imitation learning on a physical robot [31].

Discriminative Neural Networks (NN) are frequently used in trajectory prediction tasks but rarely within the locomotion domain. In [32], an auto-encoder has been used to reconstruct the robot's state from a three-dimensional latent encoding; Given the reconstructed states, they can execute trajectory-based control. At inference time, [32] produce locomotion by injecting time dependant oscillatory latents ($\in [0, 1]$) into the decoder, enabling the generation of unseen gait patterns but not locomotion. In [33], a fully connected NN has been trained to predict trajectories given the robots' state. Despite achieving a low validation loss, functional locomotion is not observed due to the exclusion of time-dependent inputs. However, the model was trained to seed the NN of an RL agent, where functionality was not the primary concern. Generative models recently proposed have shown greater effectiveness. In [34], a cVAE has been used to develop a controller capable of navigating obstacles, gaps, and other challenging terrains. VAE-Loco [35] uses a disentangled VAE [36] for trajectory prediction, producing an omnidirectional controller that controls the step height, frequency, and stance duration. However, as these methods are stochastic, they are not considered as a candidate solution.

In this paper, we approach the problem by providing deterministic, controllable, and learned priors, and thus bridge the gaps described in the aforementioned three ResL groups. Our core contribution is a novel ResL framework that is both sample efficient and highly controllable, providing omnidirectional locomotion at continuous velocities. Moreover, our framework is validated to be more robust and versatile than optimization-based controllers, and demonstrates considerably better performance in navigating across highly challenging terrains and robust responses to large perturbations.

The remainder of this paper is organized as follows: Section II presents the proposed methodology. In Section III, a set of simulation environments for training and evaluating the framework will be designed. Following, Section IV conducts experiments to evaluate the performance of the proposed approach, discusses the findings, and compares the framework to other approaches. Finally, Section V concludes the core findings, weaknesses, and future research directions.

II. Methodology

The locomotion framework proposed here enables omnidirectional locomotion, and demonstrates agile and versatile navigation across a broad range of unseen terrains. Given a target location, the controller must autonomously navigate a robot across challenging terrains, such that the distance D_{target} between the robot's position and the target is less than a minimum threshold D_{min}; maximizing the targets reached within a time limit. The following subsection presents a overview of our framework, and describes the non-parametric modules followed by technical details of control priors and the residual learning formulation.

A. Overview of the Proposed Architecture

The overall architecture of the proposed framework is depicted in Fig. 1. As shown, it contains a *kernel*, a *residual RL agent*, and a *PD controller*. The kernel is an MLP trained to replicate the trajectories produced by a model-based MPC controller. Given a set of velocity commands, it outputs foot target positions in cartesian space relative to the robot's base. The RL agent learns to generate residual positional trajectories, learning the robot's dynamics and skills, such as balance recovery, providing agility and versatility to the framework. It produces foot target position deltas, summed with the kernel output to retrieve the final targets, as shown to be most effective by [18]. The final foot target positions are converted into target joint angles using inverse-kinematics. The PD controller is responsible for generating the applied joint torques to realize the target joint angles.

B. Analytical Components

Command Generator: We introduce a Command Gerneator module that generates X-Y and yaw velocity commands, given the robot's current location, pos_{base}, and orientation, orn_{base}, for chasing after a randomly sampled target location, pos_{target}. The commands update at a frequency of

20 hz, with a maximum delta of ± 0.005. Velocity commands are constrained with in the range X:± 0.5, Y: ± 0.2, Yaw: $\pm \pi/4$.

Gait Generator: The gait generator, inspired by [31] and [37], produces a contact schedule according the internal parameters: leg phases $\phi_{1:4} \in (0, 1]$, initial phases $\theta_{1:4} \in (0, 1]$, swing ratio $r_{swing} \in (0, 1]$, and stance duration τ_{stance}. $\phi_{1:4} \in (0, 1]$ are updated at each time-step (200hz). Step cycles consist of two states: *stance* ($\phi_i > r_{swing}$), when the feet are in contact with the ground, and *swing* ($\phi_i \leq r_{swing}$) when not. Given the initial phases and the current time, we calculate the current phases:

$$\tau_{swing} = \tau_{stance}/(1 - r_{swing})r_{swing}, \quad (1)$$

$$\tau_{step} = \tau_{stance} + \tau_{swing}, \quad (2)$$

$$\phi_i = \theta_i + (\tau/\tau_{step}) \ mod \ 1. \quad (3)$$

Different gaits are mainly defined by $\theta_{1:4}$, which determine the coordination between legs. When using the MPC controller [31], τ_{stance} and r_{swing} must be tuned to produce feasible gait patterns. Gait parameters are in Table I.

TABLE I: Gait generator parameters for different gaits.

Gaits	θ_1	θ_2	θ_3	θ_4	τ_{stance}	r_{swing}
walk	0.	0.5	0.75	0.25	0.3	0.25
trot	0.9	0.4	0.4	0.9	0.3	0.4
bound	0.4	0.4	0.9	0.9	0.1	0.3

PD controller: We apply the torque control loop at 1000hz, as shown to be effective in the prior work of MELA [38]. The K_p and K_d parameters are in Table II.

TABLE II: Parameters of the PD controller.

Gains	abductor	hip	knee
K_p	100	100	100
K_d	1	2	2

Low Pass Filter: As in [19], only the residual outputs of the agent, $\delta\mathcal{A}_t$, are parsed by the LPF as the kernel trajectories are feasible and smooth:

$$\delta\mathcal{A}_t^{lpf} = \alpha\delta\mathcal{A}_t + (1 - \alpha)\delta\mathcal{A}_{t-1}^{lpf}, \quad (4)$$

where α is the smoothing factor, $\delta\mathcal{A}_t^{lpf}$ is the residual after passing through the LPF. Here setting α=0.1 can sufficiently remove most of the noise. Some noise is beneficial for policy exploration and improves responsiveness during highly noisy instances where over-smoothing introduces bias.

C. Kernel

Training Labels: During swing states, the MPC controller [39], [31], [37] uses Raibert Heuristics [40], which generates positional target trajectories p_{swing}^{ref}. We use these as labels for the swing legs. During stance states, we use the foot positions p'_{stance} after applying the motor torques generated by the MPC stance controller as the labels.

Network Inputs include the leg phase variables and velocity commands. We use the transformed normalized phase $|\phi_i|$ (5), forcing the phase greater than one during swing states; This differentiates swing and stance states in input space while allowing the network to generalize to different gaits using an alternative r_{swing}.

$$|\phi_i| = \begin{cases} 1 + (\phi_i/r_{swing}), & \text{if } \phi_i <= r_{swing} \\ (\phi_i - r_{swing})/(1 - r_{swing}), & \text{otherwise} \end{cases}. \quad (5)$$

We denote the previously described as *kernel-base*. As it accepts all the leg phases, it models the relative leg phases; As such, it cannot predict alternative gait patterns. *kernel-ind* overcomes this modeling each leg individually, passing a single leg phase, velocity commands, and a one-hot encoding referring to the target leg. The final variant, *kernel-ext*, builds on kernel-base, additionally accepting step height and ride height commands. Note, in the data collection process for *kernel-ext*, we randomly select either step height $\in [0.05, 0.18]$ (default: 0.1) or the ride height $\in [0.18, 0.28]$ (default: 0.24) before walking to a new target location.

Using Optuna [41] to perform hyper-parameter tuning with Bayesian Optimization, we find the best results using the hyper-parameters in Table III.

TABLE III: Kernel hyper-parameters (All variants).

LR	Linear LR decay	Dropout	Batch-norm
0.0024,	0.7	$5e-6$	False
Network	Activation	Loss	Batch size
(256x4)	ReLU	L1	200

D. Residual Agent

The residual RL agent, outputs positional residuals with a maximum magnitude of 5cm in each dimension for each leg. We also note that the kernel-base variant is applied for these experiments. Table IV shows the PPO hyper-parameters selected via a random search.

TABLE IV: PPO hyper-parameters.

LR	LR exp decay	Entropy	Epochs	Rollout
$1e-3$	$1e-7$	$5e-6$	10	20000
Batch size	FE	Actor	Critic	
4000	(128x2)	(128x1)	(641)	

State space: As opposed to other ResL methods providing deterministic priors [17], [18], we find excluding the reference motion results in better learning. Although, we found improved performance passing the leg phases variables. Peak performance was achieved including neither, but passing the residual after passing through the LPF from the previous time-step δA_{t-1}^{lpf}, which rectifies the Markov Property violation induced by using a LPF.

TABLE V: Residual RL agent state features.

Feature	Description	Dimensions
v_{base}	Frontal, lateral, vertical velocities of the robots base	3
a_{base}	Roll, pitch, yaw velocities of the robots base	3
v_{cmd}	Target frontal and lateral velocities	2
a_{cmd}	Target yaw velocities	1
q	Joint angles	12
$\dot{q}$	Joint angles rotational velocities	12
CoM	Position of the center of the mass	3
$pitch_{base}$	Pitch of the robot base	1
$roll_{base}$	Roll of the robot base	1
$fc_{1:4}$	Contact state of each foot of the robot	4
δA_{t-1}^{lpf}	Residual applied at the previous time-step	12

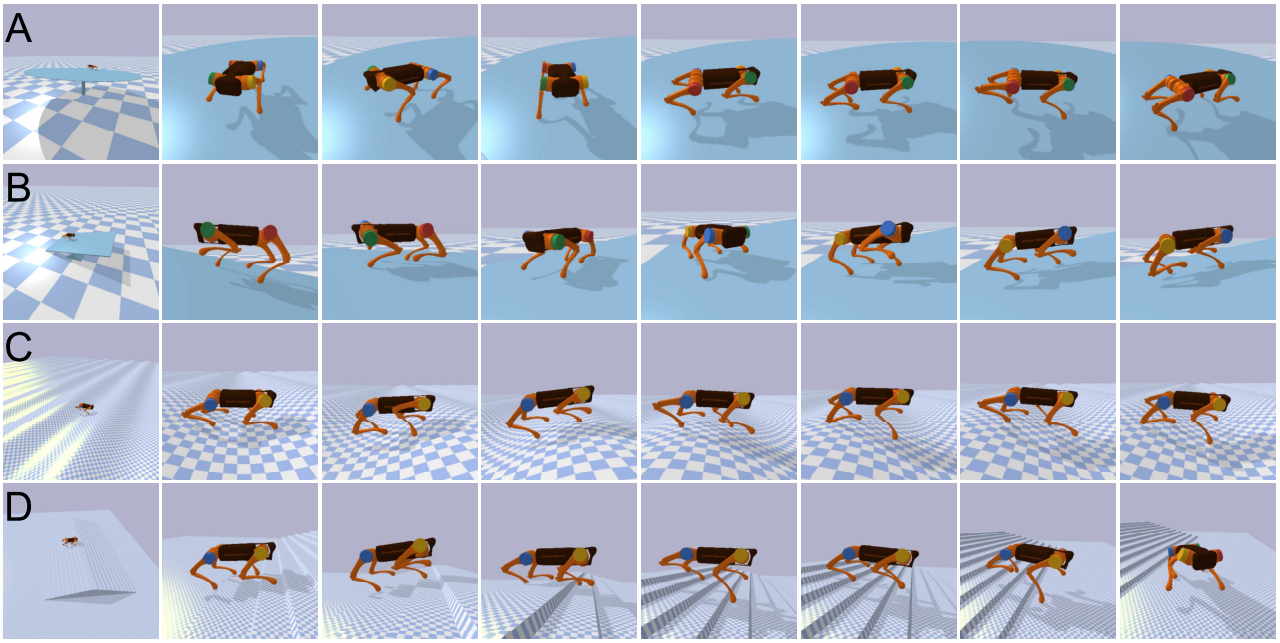

Fig. 2: Evaluation of zero-shot task generalization on different terrains: (A) Tabletop: a 360-degrees see-saw platform with the maximum inclination angle of 5 degrees; (B) A seesaw table with maximum 6 degrees inclination angle; (C) sinusoidal surface; (D) Stairs on a flat ground.

TABLE VI: Reward function parameters.

$\phi_i \in \Phi$	γ'	γ	q	weight
Linear velocity	v_{cmd}	v_{base}	18.42	0.0076
Angular velocity	a_{cmd}	a_{base}	7.47	0.0264
Center of mass	$[0, 0, -1]$	CoM	2.35	0.0298
Distance to target	0	D_{target}	0.74	0.0169
Roll and Pitch	$[0, 0]$	$[pitch_{base}, roll_{base}]$	7.47	0.0298
$r_i \in \mathcal{F}_{nom}$	Reward function			weight
Falling penalty	$r = \begin{cases} -19.8, & \text{if the robot fell} \\ 0, & \text{otherwise} \end{cases}$			1
Target reached	$r = \begin{cases} 8.75, & \text{if } D_{target} \leq D_{min} \\ 0, & \text{otherwise} \end{cases}$			1

Reward Function: We use a mixture of radial basis functions (RBF), $\phi_i(\gamma', \gamma, q) = \exp(-(\gamma' - \gamma)^2 q)$, (shown to be effective in [30], [38]), and nominal rewards $r_i \in \mathcal{F}_{nom}$ to define each feature of the reward function. RBF reward function features, $\phi_i \in \Phi$, are parameterized by the target, γ', and the curve steepness, q; A steeper RBF function incentivises learning and accommodates for attributes with small numeric errors. Equation (6) represents our reward function and its parameters are summarized in Table VI.

$$R_t = \sum_{\phi_i \in \Phi} \omega_i \phi_i(\gamma_i', \gamma_i, q_i) + \sum_{r_i \in \mathcal{F}_{nom}} r_i. \tag{6}$$

III. Simulations

In this section, multiple scenarios for different aspects of training and evaluation of the kernel and residual agent will be designed. To this end, a simulated A1 Unitree quadruped is used in the PyBullet [42] physics simulator. In our simulations, we wrap the PyBullet simulation in an Open-Ai Gym [43] environment during RL experiments.

A. Training the framework

The first stage of the training process, training the kernel, requires collecting locomotion data from an expert MPC controller. The MPC controller [31] executes the trot gait, with a stance duration of 0.2 s, which reduces the variation in CoM allowing the network to learn better. It navigates to 500 consecutive target locations over the flat terrain, set at a minimum distance of 2.5m in a random direction. Collecting the data network inputs $\{ v_{cmd}, a_{cmd}, q, \phi_{1:4}\}$ before actions are taken, and labels $\{p^{ref}_{swing}, p'_{stance}\}$ and after each time-step (200hz).

In the second stage, we train the residual RL agent on randomly selected terrains for five consecutive episodes (75% height field, 25% perlin). The height-field perturbations are sampled uniformly $\sim\in [3\text{cm}, 4.5\text{cm}]$. Also, force perturbations are applied to the robot at a random point on the robot body, in a random direction horizontally, at intervals $\sim\in [5, 8]$ seconds, with a magnitude $\sim\in [100, 350]$N, for a duration of 0.3s. The agent is trained for a total of 20M timesteps, tasked with navigating to randomly selected target locations, with a precision of $D_{min} = 0.5m$, using 5 cpu's in parallel, taking roughly 8 hours (NVIDIA GeForce GTX 1060 6GB, AMD Ryzen 5 2600X Six-Core Processor).

B. Evaluating the framework

The framework is evaluated for it's versatility in four terrains in ascending difficulty: A) Tabletop, B) Seesaw, C) Sinusoidal, and D) Stairs (see Fig. 2). We set 5 target locations to reach per run, placed to challenge the agent, and start from 4 different starting locations. The pivoting tabletop has a maximum rotation around the pivot of 5deg.

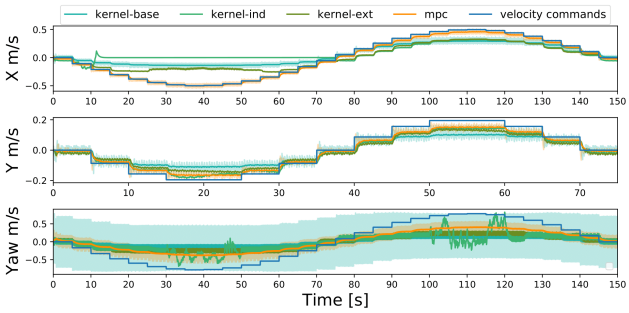

Fig. 3: The realized velocities of the robot given velocity commands for the MPC controller and kernel variants.

The seesaw has an decline/incline of 6deg. The stairs have a step height of 4cm. The sinusoidal terrain has a maximum incline of 11.5deg.

Furthermore, we separately evaluate the robustness of the framework applying external forces to the robot. It is tasked with walking to a single target location on a flat terrain, where a force is applied to a random location on the robots body in a random direction in the horizontal plane for a duration of 0.3 seconds. We determine success by the robots ability to reach the target location. For each magnitude of force applied ([250N, 900N]), we run 10 attempts and record the percentage of successfully completed tasks, as shown in Table X detailed in the next section.

IV. Results and Analysis

The section analyzes the experiments, discusses observations in relation to related works, and provides numerical evaluations for the kernel and the framework as a whole.

A. Kernel Analysis

To understand the degree kernel variants capture the characteristics and controllability of the MPC controller, we compare the velocity control performance exhibited on a flat terrain, where a single velocity command is varied while the others are fixed to zero. Fig. 3 demonstrates a performance gap between all variants and the MPC controller. The kernels cannot move at negative frontal velocities nor can they match the maximum lateral, angular and positive frontal velocities achieved with the MPC controller. In addition, the kernels experience extremely high variance when turning, showing a significant performance gap in the realized yaw velocities. We observe no performance deterioration in *kernel-ext* from *kernel-base*, despite achieving lower validation loss (Table VII). *kernel-ind* is the weakest when moving at negative frontal velocities, but also experiences erratic behaviour when commanded with high yaw velocities.

The variant, kernel-ind, demonstrates gait generalization capabilities, producing unseen gait patterns that result in effective locomotion. Fig. 4 shows the production of walk and bound gaits, which were not provided during training. Although it produces these gaits, the kernel behaves undesirably when inputting high yaw commands as seen when executing the trot gait Fig. 3.

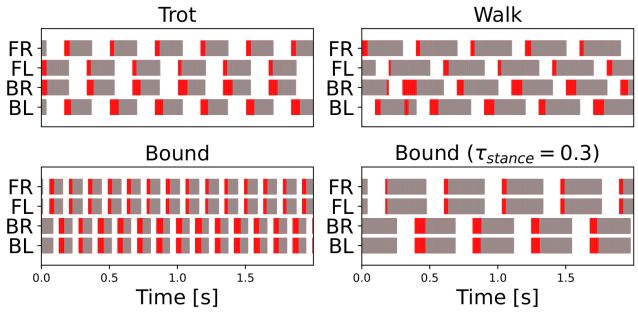

Fig. 4: Zero-shot gait patterns generated using Kernel-ind. Grey segments show the realized foot contacts, while the red segments show foot contact error against the contact schedule from the gait generator.

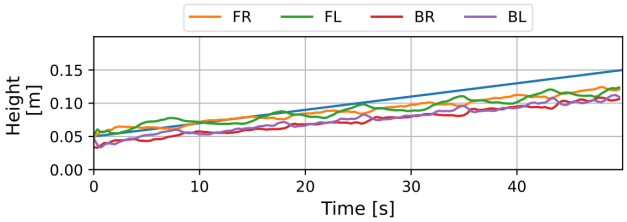

Fig. 5: The peak realized the height of each foot over a step cycle as the step height command increases (blue line), controlled using kernel-ext.

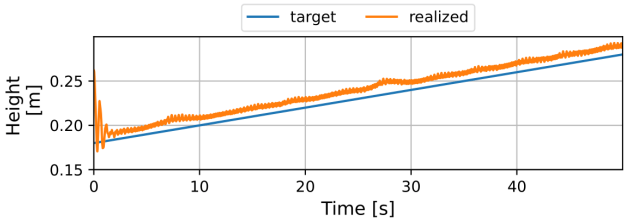

Fig. 6: The realized height of the robots' base as the ride height command increases (blue line), controlled using kernel-ext.

Training of *kernel-ext* results a minimal increase in the validation loss (L1=$7.1e-4$, see Table VII), while allowing us to control the ride and step heights live, as shown in Fig. 5 and Fig. 6 (The video can be found at **https://youtu.be/bUZJadWCRXU**). We observed that the target step height and ride height commands are not realized precisely, although it clearly demonstrates the desired behaviour. Furthermore, we see greater inaccuracies in the realized steps heights, where the error increases as the target height increases.

B. Kernel Results

Our method demonstrates far superior results ($6.2e-4$ mean absolute error) compared to [33], which achieves a validation loss 0.007 (MSE), approximating 0.083 mean absolute error. Furthermore, our method yields a functional locomotion controller, as demonstrated by Fig. 3. The results (Table VII) required training on 2.1 hours of locomotion data. Table VIII shows the results of training with less data, determined by the number of target locations reached. The validation performance deteriorates as the number of targets decreases. However, training with only ten target locations

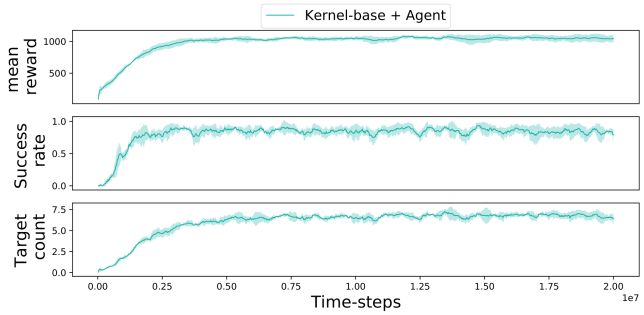

Fig. 7: Training of the agent with *kernel-base* to provide the priors, with the mean and standard deviation over four seeds.

TABLE VII: Performance of kernel variants, showing the mean minimum validation loss and the standard deviation.

Kernel-variant	Mean Validation Loss	Standard Deviation
Kernel-base	$6.2e-4$	$6.9e-6$
Kernel-ind	$7.2e-4$	$4.9e-6$
Kernel-ext	$7.1e-4$	$1.0e-5$

TABLE VIII: Kernel-base performance as the amount of data increases.

Number of Targets	10	25	50
Mean Validation Loss	$1e-3$	$9.1e-4$	$8.4e-4$
Standard Deviation	$3.1e-6$	$9.8e-6$	$5.3e-6$
Number of Targets	100	200	400
Mean Validation Loss	$7.7e-4$	$6.7e-4$	$6.2e-4$
Standard Deviation	$6.2e-6$	$5.5e-6$	$6.9e-6$

(3.1 minutes), the kernel achieves a validation loss of $1e-3$, capable of producing locomotion simulation.

C. Residual Agent Analysis

During training, we record the success rate, target count, and reward. An episode is considered successful after navigating to more than two target locations and not falling. The target count is the number of target locations reached with in a 60s period. The agent converges after only 7.5M timesteps (see Fig. 7), showing significantly improved sample efficiency over other omnidirectional ResL methods: [20], [19], and [23], requiring 250M, 100M, and 200M timesteps, respectively. This suggests deterministic reference motions, as provided by kernel-base and gait libraries, simplifies the learning scenario. Furthermore, our framework outperformed the kernel to seeded-agent framework [33], which required 200M timesteps.

D. Residual Agent Results

We measured the average reward per time-step, using the final reward function (Table VI); The success rate, defined as the proportion of complete runs (reaching all the targets), and the fall rate. Our framework (kernel+agent) demonstrates versatility outperforming the MPC controller used to train the kernel in every evaluation terrain with a success rate of 93% in the most challenging stairs terrain. The results are summarized in Table IX. Furthermore, it is more robust against perturbations, able to regularly recover its balance

TABLE IX: Evaluation comparing locomotion controllers.

Tabletop	**Reward/steps**	**Num Targets**	**Success Rate**
MPC (0.2)	0.016±0.11	2.5±2.89	0.5
Kernel	0.065±0.0085	2.25±1.5	0
Kernel+Agent	0.097±0.001	5±0.0	1
Seesaw	**Reward/steps**	**Num Targets**	**Success Rate**
MPC (0.2)	0.065±0.018	0.0±0.0	0
Kernel	0.043±0.00245	0.0±0.0	0
Kernel+Agent	0.091±0.0006	5±0.0	1
Stairs	**Reward/steps**	**Num Targets**	**Success Rate**
MPC (0.2)	0.073±0.0022	0.0±0.0	0
Kernel	0.047±0.0019	0.0±0.0	0
Kernel+Agent	0.089±0.0024	4.75±1.0	0.9375
Sinusoidal	**Reward/steps**	**Num Targets**	**Success Rate**
MPC (0.2)	0.082±0.0057	3±1.83	0.25
Kernel	0.042±0.0021	0±0.0	0
Kernel+Agent	0.089±0.0026	4.75±1.0	0.9375

TABLE X: Robustness against perturbations, using the MPC with (τ_{stance} =0.2).

Force (N)	250	300	350	400	450	500	550
MPC	1.0	0.8	0.9	0.7	0.5	0.3	0.4
Kernel	1.0	1.0	1.0	0.8	0.5	0.3	0.3
Kernel+Agent	1	1	1	1	0.9	0.8	0.9
Force (N)	600	650	700	750	800	850	900
MPC	1.0	0.4	0.1	0.2	0	0	0
Kernel	0	0	0	0	0	0	0
Kernel+Agent	0.8	0.7	0.4	0.8	0.6	0.3	0.2

after perturbations of 800N where the MPC controller fails (Table X).

V. Conclusions

In this work, we developed a ResL framework that is both sample efficient and highly controllable, providing omnidirectional locomotion at continuous velocities. We achieved this by providing deterministic trajectory priors using a NN trained on expert data collected from an MPC controller. Additionally, our residual agent applied positional trajectories without knowledge of the priors or the terrain. Through a set of simulated scenarios, the framework demonstrated navigation on the most challenging terrains and demonstrated superior performance over the MPC controller used to train the kernel. Furthermore, the kernel exhibited gait generalization capabilities, producing locomotion for walk and bound gaits, when provided with only trot data.

For future work, we propose using the residual agent to adapt the trajectories produced for unseen gaits, to enable expert level control without any guidance directly from an expert controller. Additionally, we hypothesise the framework could exhibit greater robustness if the agent has direct control over the body height and the step height.

Acknowledgement

This work is partially supported by EU H2020 project Enhancing Healthcare with Assistive Robotic Mobile Manipulation (HARMONY, 101017008).

References

[1] J. Bhatti, A. R. Plummer, P. Iravani, and B. Ding, "A survey of dynamic robot legged locomotion," in *2015 International Conference on Fluid Power and Mechatronics (FPM)*, 2015, pp. 770–775.

[2] M. F. Silva and J. T. Machado, "A historical perspective of legged robots," *Journal of Vibration and Control*, vol. 13, no. 9-10, pp. 1447–1486, 2007. [Online]. Available: https://doi.org/10.1177/1077546307078276

[3] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, p. eabc5986, 2020. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abc5986

[4] 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)*, 2018, pp. 1–9.

[5] B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6295–6301.

[6] 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.

[7] I. Chatzinikolaidis, Y. You, and Z. Li, "Contact-implicit trajectory optimization using an analytically solvable contact model for locomotion on variable ground," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6357–6364, 2020.

[8] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," 2019.

[9] F. Abdolhosseini, H. Y. Ling, Z. Xie, X. B. Peng, and M. van de Panne, "On learning symmetric locomotion," in *Motion, Interaction and Games*, ser. MIG '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3359566.3360070

[10] D. R. Song, C. Yang, C. McGreavy, and Z. Li, "Recurrent deterministic policy gradient method for bipedal locomotion on rough terrain challenge," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*. IEEE, 2018, pp. 311–318.

[11] W. Yu, G. Turk, and C. K. Liu, "Learning symmetric and low-energy locomotion," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–12, aug 2018. [Online]. Available: https://doi.org/10.1145%2F3197517.3201397

[12] C. Yang, K. Yuan, S. Heng, T. Komura, and Z. Li, "Learning natural locomotion behaviors for humanoid robots using human bias," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2610–2617, 2020.

[13] 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," 2018. [Online]. Available: https://arxiv.org/abs/1804.10332

[14] S. Grillner and P. Wallén, "Innate versus learned movements—a false dichotomy?" in *Brain Mechanisms for the Integration of Posture and Movement*, ser. Progress in Brain Research. Elsevier, 2004, vol. 143, pp. 1–12. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S007961230343001X

[15] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," 2018. [Online]. Available: https://arxiv.org/abs/1812.03201

[16] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," 2019. [Online]. Available: https://arxiv.org/abs/1903.11239

[17] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne, "Feedback control for cassie with deep reinforcement learning," 2018. [Online]. Available: https://arxiv.org/abs/1803.05580

[18] H. Duan, J. Dao, K. Green, T. Apgar, A. Fern, and J. Hurst, "Learning task space actions for bipedal locomotion," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1276–1282.

[19] M. Kasaei, M. Abreu, N. Lau, A. Pereira, and L. P. Reis, "A cpg-based agile and versatile locomotion framework using proximal symmetry loss," 2021. [Online]. Available: https://arxiv.org/abs/2103.00928

[20] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Real-time trajectory adaptation for quadrupedal locomotion using deep reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 5973–5979.

[21] H. Shi, B. Zhou, H. Zeng, F. Wang, Y. Dong, J. Li, K. Wang, H. Tian, and M. Q. H. Meng, "Reinforcement learning with evolutionary trajectory generator: A general approach for quadrupedal locomotion," 2021. [Online]. Available: https://arxiv.org/abs/2109.06409

[22] C. Yu and A. Rosendo, "Multi-modal legged locomotion framework with automated residual reinforcement learning," 2022. [Online]. Available: https://arxiv.org/abs/2202.12033

[23] W. Jungdam, G. Deepak, and H. Jessica, "Physics-based character controllers using conditional vaes," *ACM Transactions on Graphics (SIGGRAPH 2022)*, 2022.

[24] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf

[25] A. Sprowitz, M. ajallooeian, A. Tuleu, and A. Ijspeert, "Kinematic primitives for walking and trotting gaits of a quadruped robot with compliant legs," *Frontiers in Computational Neuroscience*, vol. 8, 2014. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fncom.2014.00027

[26] A. Singla, S. Bhattacharya, D. Dholakiya, S. Bhatnagar, A. Ghosal, B. Amrutur, and S. Kolathaya, "Realizing learned quadruped locomotion behaviors through kinematic motion primitives," 2018. [Online]. Available: https://arxiv.org/abs/1810.03842

[27] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 2, 2002, pp. 1398–1403 vol.2.

[28] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," 2021. [Online]. Available: https://arxiv.org/abs/2102.03861

[29] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning rhythmic movements by demonstration using nonlinear oscillators," in *Proceedings of the ieee/rsj int. conference on intelligent robots and systems (iros2002)*, no. CONF, 2002, pp. 958–963.

[30] T. Li, J. Won, S. Ha, and A. Rai, "Fastmimic: Model-based motion imitation for agile, diverse and generalizable quadrupedal locomotion," 2021. [Online]. Available: https://arxiv.org/abs/2109.13362

[31] X. B. Peng, E. Coumans, T. Zhang, T.-W. E. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," in *Robotics: Science and Systems*, 07 2020.

[32] H. Yamamoto, S. Kim, Y. Ishii, and Y. Ikemoto, "Generalization of movements in quadruped robot locomotion by learning specialized motion data," *ROBOMECH Journal*, vol. 7, no. 1, pp. 1–14, 2020.

[33] A. Li, Z. Wang, J. Wu, and Q. Zhu, "Efficient learning of control policies for robust quadruped bounding using pretrained neural networks," 2020. [Online]. Available: https://arxiv.org/abs/2011.00446

[34] D. Surovik, O. Melon, M. Geisert, M. Fallon, and I. Havoutis, "Learning an expert skill-space for replanning dynamic quadruped locomotion over obstacles," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 1509–1518. [Online]. Available: https://proceedings.mlr.press/v155/surovik21a.html

[35] A. L. Mitchell, W. Merkt, M. Geisert, S. Gangapurwala, M. Engelcke, O. P. Jones, I. Havoutis, and I. Posner, "Vae-loco: Versatile quadruped locomotion by learning a disentangled gait representation," 2022. [Online]. Available: https://arxiv.org/abs/2205.01179

[36] E. Mathieu, T. Rainforth, N. Siddharth, and Y. W. Teh, "Disentangling disentanglement in variational autoencoders," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4402–4412.

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

[38] C. Yang, K. Yuan, Q. Zhu, W. Yu, and Z. Li, "Multi-expert learning of adaptive legged locomotion," *Science Robotics*, vol. 5, no. 49, dec 2020. [Online]. Available: https://doi.org/10.1126%2Fscirobotics.abb2174

[39] Erwincoumans, "Code accompanying the paper learning agile robotic locomotion skills by imitating animals." [Online]. Available: https://github.com/google-research/motion_imitation

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

[41] "A hyperparameter optimization framework." [Online]. Available: https://optuna.org/

[42] Admin, "Bullet real-time physics simulation," Mar 2022. [Online]. Available: https://pybullet.org/wordpress/

[43] "Gymlibrary.ml." [Online]. Available: https://www.gymlibrary.ml/