Realizing Linear Controllers for Quadruped Robots on Planetary Terrains

Aditya Shirwatkar[1], Somnath Kumar[1], Bharadwaj Amrutur[1], Shalabh Bhatnagar[1], Ashitava Ghosal[1], Shamrao[2], Vinod Kumar[2], Shishir Kolathaya[1]

***Abstract*— Until now, planetary exploration has been accomplished with wheeled vehicles, which makes movement in highly complex, sandy, and sloping terrain incredibly tough. On the other hand, legged robots have come a long way in the last decade and have reached a stage of development where practical applications appear to be possible. Legged robots can overcome the difficulties wheeled vehicles face when exploring harsh environments like impact craters to collect critical scientific data. As a result, there is a need to develop simple, stable walking controllers given the limited power resources and reserve maximum onboard compute for scientific equipment while exploring such regions. This work proposes a walking controller for legged robots that is computationally efficient at runtime for traversing planetary terrains. We realize this walking controller on our quadruped Stochlite, using learned linear feedback policies that modulate the end-foot trajectories. The proposed walking controller can traverse on various planetary terrains such as flat, sloped, rugged, loose, and lower-than-Earth gravity conditions in simulation environments. Our controller outperforms the baseline open-loop controller on such planetary terrains by reducing the slippage and increasing the stability. In addition, we have also provided preliminary hardware testing results of our controller.**

Keywords: *Quadrupedal walking, Reinforcement Learning, Random Search, Planetary Terrains*

I. Introduction

Future robotic exploration missions into our solar system will face more challenging terrain. Extreme habitats in breakouts and exposed bedrock, such as those seen in craters or along cliffs, provide information about the planet's geological history. Researchers are interested in researching this wide range of geological locations that are extremely difficult or impossible to reach with standard wheel vehicles because of the highly distributed surface, steepness, or unpredictability in terrain characteristics. Extreme habitats in breakouts and exposed bedrock, such as those seen in craters or along cliffs, provide information about the planet's geological history. Simply put, the greater the variety of the environment, the greater the scientific potential.

Because of the unpredictability of the encountered soil and the requirement to traverse mountainous terrain, a wheeled vehicle may be the limiting factor in such a mission. Mobile robotic exploration of the Moon and Mars has previously relied only on wheeled locomotion. Such systems provide exceptional stability and robustness on flat terrain, but their operating range is limited [1]. However, these wheeled vehicles hit their limits in unstructured areas with high, sandy slopes and rough terrain patches [2]. Dynamic legged robots have been demonstrated to walk efficiently on challenging terrains [3]. The maturity of the technology has allowed the robots to transition from the lab to real-world applications [4].

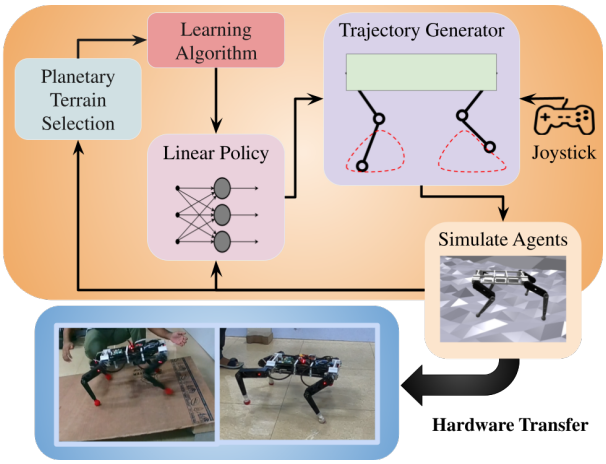

Fig. 1: Overview of Complete Pipeline

Classical works such as the spring-loaded inverted pendulum (SLIP) with Raibert's heuristic controller [5] and the Zero Moment Point (ZMP) [6] approach have been used to generate robust walking behaviors. Convex Model Predictive Control (MPC) [7] approach has shown dynamic motion behaviors using only the centroidal dynamics model. Nevertheless, these optimization-based methods suffer from the drawback of requiring heavy onboard compute resources and relying on knowing the system's dynamics model.

Concurrently, techniques for walking robots based on Reinforcement Learning (RL) have gained prominence. RL combined with the centroidal dynamics model of quadruped has been shown to solve stepping-stone locomotion, two-legged in-place balance, and balance beam locomotion with a good sim-to-real transfer [8]. There have also been approaches that use Deep RL policies to parameterize end-effector foot trajectories which have shown excellent results in the real world [9].

These current State of the Art controllers prove that legged robots have immense potential in real-world applications. Robots like ATHLETE [10], Scorpian [11], SpaceClimber [12], and SpaceBok [13] have been explicitly developed

This project is supported by the Indian Space Research Organization under grant no. STC/ECS/SIV459.

[1]Robert Bosch Centre for Cyber Physical Systems, IISc, Bengaluru (email: stochlab@iisc.ac.in)

[2]U R Rao Satellite Centre, ISRO, Bengaluru

for planetary exploration. ATHLETE, Scorpian, and Space-Climber offer good stability while traversing rough terrain but have limitations on hardware that acts as a bottleneck for traversability speed and agility. SpaceBok, on the other hand, offers excellent dynamic maneuvers, which have been shown to work in low gravity conditions [14] and sandy terrains [15]. However, considering the limited resources available for space exploration missions, having computationally efficient and robust controllers is critical, as most of the compute should be accessible to the scientific equipment. Linear feedback policies have shown promising alternatives to the classical model-based, and Deep RL approaches [16], [17]. These linear policy-based approaches allow one to realize an efficient and robust controller for walking robots.

Motivated by these findings, we propose a control framework consisting of linear feedback policies to generate walking trajectories on flat, sloped, rough, loose (soiled/sandy), and lower-than-Earth gravity environments such as Mars and Moon. The following points can summarize our main contributions in this work:

- Learning linear policies to control a highly non-linear system such as a quadruped in various planetary terrains and gravity conditions.
- Large scale approximate terramechanics subroutine for quadruped robots in simulation.
- Preliminary hardware testing to showcase the capabilities of our controller.

Our approach significantly differs from [14] and [15] as we neither use Deep Neural Network (DNN) policy nor optimization-based control methods. This allows us to have a computationally efficient and interpretable controller. To the best of our knowledge, there is no large-scale real-time solution for terramechanics simulation which can be readily integrated with learning algorithm pipelines. As a result, we utilized Nvidia's IsaacGym [18] in conjunction with a Bekker model's subroutine [19] to mimic sand-soil contact interactions. It is worth mentioning that we did not attempt to focus on sinkage but rather on slippage analysis. Finally, we also demonstrate a hardware transfer of our controller, showing an improvement over open-loop controllers in terms of sheer robustness.

The following is how the paper is organized: Section II will go through the robot model, notations, and hardware specifications. Section III describes the linear policy, the walking controller, and terramechanics subroutine details. Section IV describes the training and evaluation methods used. Section V showcases a summary of the simulation results, assessments, comparison with the baseline open-loop controller, and preliminary hardware tests. Finally, Section VI is the conclusion of this work.

II. Robot Description

This section will briefly explain the robot used to showcase the proposed methodology and the accompanying actuator-sensor, terrain orientation estimation, and kinematics framework we intend to employ for locomotion.

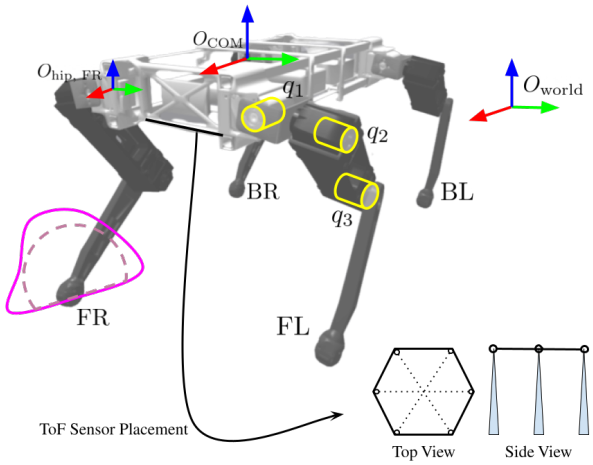

Fig. 2: Figure showing Stochlite's kinematic description where dashed curve shows the ideal end-foot trajectory generated by the open-loop controller, and bold curve in pink shows the modulation of the ideal trajectory by the linear policy

Stochlite, as shown in Fig. 2, is a custom-built low-cost quadruped robot developed for rapid prototyping of learning-based controllers. The symbols FL, FR, BL, BR, represent the front-left, front-right, back-left, and back-right legs respectively. Also, O_{world} and O_{COM} represents world and Centre of Mass (COM) frames respectively. For simplicity, only the FR hip frame, $O_{\text{hip, FR}}$ is shown. It is worth noting that, unlike SpaceBok [13], the quadruped is not explicitly built for planetary exploration applications, as in this work, we want to focus on building the model-free control framework that is easily transferable to any quadruped system.

1) Actuation: Overall, the robot model consists of 6 floating and 12 Degrees of Freedom (DoF). The 12 DoF are actuated using B3M Smart Servo motors. The motors can measure joint angles (hip and knee) and torques via joint encoders and motor current sensors. The Stall torque provided by the motors is in the range of $4.1\ \mathrm{N\,m}$

2) Onboard Computation: We use a Raspberry PI 3b microcomputer for running all the high-level controllers. It consists of 2 GB RAM and includes four high-performance ARM Cortex-A53 processing cores running at 1.2 GHz. An STM32 microcontroller board is also used for low-level communication between a Raspberry Pi and the servo motors using Serial Peripheral Interface (SPI) communication.

3) Sensors: Accurate feedback is necessary for any controller to drive the system to the desired state. For this, we use an Xsens MTi-610 Inertial Measurement Unit (IMU), which provides calibrated data on the 3D orientation, angular velocities, acceleration, and magnetic field.

4) Local Terrain Slope Estimation: We require the local terrain slope as feedback to the controller. Hence we use a similar technique as [16] to estimate it with the help of joint encoders in the motors and six Time of Flight (ToF) sensors placed in a hexagonal configuration below the torso as shown

in Fig. 2. Note ToF sensors are only used to estimate the readings on hardware as the current version of IsaacGym does not support range sensors. This ToF sensor placement is required as currently, we lack any foot force sensing capabilities for contact detection and thus are planned for the future versions of the robot.

5) Kinematics: We treat each leg independently to derive analytical relations for forward and inverse kinematics. Here q_1, q_2, and q_3 represent abduction, hip, and knee joints and form a serial-3R kinematic chain as shown in Fig. 2. This reduces the runtime overhead present in the iterative solvers, as it is one of the critical components that help us map end-foot trajectories to joint space.

III. Methodology

This section provides an overview of the control architecture, i.e., how end-foot trajectories are generated and tracked in real-time to walk in various environments. This structure is also depicted graphically in Fig. 1.

A. Reinforcement Learning

The locomotion of quadruped in this work is treated as an RL problem. We parameterize our RL policy with end-foot trajectories with sinusoidal height variation to accelerate training. Our method is similar to [16], [17], which uses a feedback mechanism to dynamically alter these trajectories based on body and local terrain slope. At a high level, the RL policy infers the parameters of the walking trajectories and, therefore, robot motion. We limit our policy to being linear, as it then requires low computation, allowing our policy to be executed on an onboard embedded system in real-time. Section IV contains further information on the training algorithm used to learn these linear policies.

1) Observation Space: The observation space, which is the input to the policy, is in $\mathbb{R}^{18\times 1}$ in our formulation. It consists of robot walking height z, body orientation in roll ϕ, pitch θ, yaw ψ, the position of foot $r_i \in \mathbb{R}^3$ for each leg $i \in \{1, 2, 3, 4\}$ with respect to center of mass (COM), local terrain slope in roll ϕ_s and pitch θ_s.

2) Action Space: The action space, i.e., the output of the policy, is in $\mathbb{R}^{8\times 1}$, which represents the instantaneous shifts. The shifts s_i are x, y, z translational transforms for the leg trajectories of the robot in the body frame. These instantaneous shifts result in reactive behavior, as seen in Fig. 2.

3) Linear Policy: We choose the policy to be $\pi(\mathbf{s}) := M(\Theta)\mathbf{s}$, where $M \in \mathbb{R}^{8\times 18}$, is a matrix that maps the observations $\mathbf{s}$ to actions, and Θ represents the learnable parameters, i.e., the policy matrix elements. To simplify the problem, we consider M to be a sparse matrix by accounting for the intuitive contribution of each element in the observation space to the elements of the action space. For example, the shift in x for the leg is only affected by the body height, body pitch, slope pitch, and x coordinate of the legs. So the column element for the respective terms will be learned, with the rest always being zero. Such structures also showcase one of the advantages of having linear feedback policies as they are easier to analyze and impose intuitive heuristics than DNNs.

B. Walking Controller

Walking over varied terrains such as flat, inclined, rough, and loose brings unique problems that cannot be acquired straight from the open-loop walking controller (our framework without the linear policy feedback). As mentioned earlier, we use the linear policy to alter the end-foot trajectories in real-time based on the observations to ensure steady walking. The trajectory of each leg is parametrized by phase $\Phi \in [0, 2\pi]$. Note that the swing phase and stance phase will be different for each leg, as they will vary according to different gaits. For the scope of this work, we focused on mainly realizing two types of gait behaviors: trot and crawl.

For the swinging trajectory, the foot placement with respect to the hip frame for each leg at the start of the swing is determined using the Raibert Heuristic [5],

$$\Delta r_{xy,i,t} = \frac{v_{cmd}\Delta T_{st}}{2} + s_{xy,i,t}. \tag{1}$$

where t is the time step, ΔT_{st} is the stance duration, v_{cmd} is the linear velocity from joystick inputs, and $s_{xy,i,t}$ are the x, y components of the shifts predicted by the linear policy. Once this is defined, the leg trajectories are generated as follows for every time step:

$$\begin{aligned}
dr_{xy,i,t} &= \begin{cases} \frac{\Delta r_{xy,i,t} - r_{xy,i,t}}{\Delta T_{sw}} \frac{\pi}{\pi - \phi_{i,t}}\, dt & \text{, when } i \text{ is swing} \\ -v_{xy,cmd}\, dt & \text{, when } i \text{ is stance} \end{cases} \\
r_{xy,i,t+1} &= r_{xy,i,t} + dr_{xy,i,t} \\
r_{z,i,t+1} &= \begin{cases} h_{sw} \sin\phi_{i,t} + h_0 + s_{z,i,t} & \text{, when } i \text{ is swing} \\ h_0 + s_{z,i,t} & \text{, when } i \text{ is stance} \end{cases}
\end{aligned} \tag{2}$$

where $\phi_{i,t}$ is the i-th leg phase, ΔT_{sw} is the swing duration, dt is the control time step, h_{sw} is the swing height, h_0 is the walking height, r_{xy} and r_z, s_z represent the x, y and z components of the foot and shifts respectively.

Thus the entire leg trajectory gets modulated by the predicted action values of the linear policy. We choose to have a decoupled structure in x, y, z for the end foot trajectory because it allows us to take twist inputs seamlessly from the joystick. In theory, the z component variation can consist of any function that generates a swinging profile; however, we used sinusoidal variation to fulfill our requirements. An inverse kinematics function then gives the command joint angles for each leg described earlier.

C. Terramechanics Subroutine Details

Controllers capable of walking over loose terrains made up of soil and sand without destabilizing the quadruped are required. Thus, policies must be explicitly trained in such environments for seamless sim-to-real transfer. Current physics simulators like Chrono [20] and Adams-Based Rover Terramechanics and Mobility Simulator (ARTEMIS) [21] lack the ability to perform real-time computations required

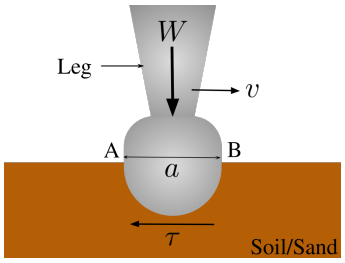

Fig. 3: Figure showing an interaction between soiled/sandy terrain and a robot leg

for fast realization of learning-based controllers. A simple alternative is to create an approximate terramechanics subroutine of the Bekker model [19] in an existing fast and large scale simulator for Robot Learning such as Nvidia IsaacGym [18]. We only focus on the shear forces acting at the contact, which causes slippage in the total distance traveled. To account for the effects of high sinkage is an exciting avenue and beyond the scope of this work; hence, it is left for future works.

1) Shear Stress at the contact: Consider the scenario shown in Fig. 3 where the contact of the leg produces a vertical load W, with a being the projected contact area of cross-section AB. For simplicity, we neglect the effects due to the curvature of the leg and treat the contact area as a flat surface. Then the pressure P, produced at the contact surface on the soil, is given by $P = W/a$. This motion creates shear stress at the cross-section AB. The Mohr-Coulomb failure criterion estimates the soil failure, which is given by,

$$\tau_{max} = C + P\ \tan(\phi_c), \tag{3}$$

where τ_{max} is the failure stress, C is the soil cohesion, and ϕ_c is the angle of friction. The actual shear stress generated is then derived as,

$$j = v\ dt, \tag{4}$$

$$\tau = \tau_{max}\ (1 - e^{-j/G}), \tag{5}$$

$$F_s = \tau\ a. \tag{6}$$

Here τ is the actual shear stress generated at the contact surface, j is the shear displacement, and G is the shear modulus. Once τ is calculated at the contact surface, one can say that a shear force F_s is acting on the body. This force then induces a slippage at contact, which means that the actual velocity of the body is reduced as it is a reaction force. Equations (3) to (6) are then used to apply an external force at the next time step of the simulation. Finally, different gravity conditions can be simulated by changing the parameters of the physics engine solver.

IV. Policy Training and Evaluation

We have used the Augmented Random Search Algorithm [22]. The algorithm is comparable to other model-free RL algorithms when searching linear deterministic policies. Given the problem setup, the algorithm's goal is to determine the parameters Θ of the matrix M that yield the best rewards, which leads to the best locomotion on different planetary terrains. To exploit the parallelization capabilities of Isaac Gym, we implemented a batched version of ARS.

A. Domain Randomization

We employ two-stage domain randomization to reduce the sim-to-real gap and produce robust learned policies. The first stage consists of randomizing the robot orientation and spawning height at the start of each episode. This ensures that the policy can recover from unstable configurations. The second stage consists of changing the terrain of locomotion using curriculum learning similar to [16]. Initially, the curriculum consisted of easier terrains like flat ground and lower slope values. After every m iterations of the learning algorithm, where m is a hyper-parameter, the difficulty of terrain is gradually increased by training on higher slope values and rough terrains. This ensures that the policy has seen all the different terrain types. While learning the linear policies, we also randomly sample the soil/sand parameters for every rollout to avoid overfitting. The properties used for our experiment are obtained from [23].

B. Reward Function

Obtaining a good reward function is critical to reduce the training time and ensure that the learned policy makes optimal predictions for the walking controller. We choose our reward function R to be,

$$\begin{aligned} R = G_{w_1,u_1}(\|v_{cmd} - v_{curr}\|) \\ + G_{w_2,u_2}(\phi - \phi_s) + G_{w_3,u_3}(\theta - \theta_s) \\ + G_{w_4,u_4}(\|\omega_{x,y}\|) + G_{w_5,u_5}(P_o). \end{aligned} \tag{7}$$

In the above equation, the function $G : \mathbb{R} \rightarrow [0, 1]$ is a Gaussian kernel and is given by $G_{w_j,u_j}(x) = w_j\ e^{-u_j x^2}$, where $j \in \{1, 2, 3, 4, 5\}, w_j$ and u_j are scalar weights. Whereas, v_{curr} and $\omega_{x,y}$ are current 2D twist and current angular velocity in x, y. The reward can be broken into five terms, encouraging the robot to minimize the error in commanded and current linear velocity, maximizing the stability by aligning the body roll and pitch to the local terrain slope, minimizing the variation in current angular velocities, and consuming less power P_o.

We have trained our robot to walk on flat, sloping grounds of up to 15° and rough terrains with an average undulation of 5 cm. The robot was commanded to track a given linear velocity in a fixed direction from emulated joystick inputs with respect to the world frame. The hyper-parameters of ARS used in training were: Learning rate $\alpha = 0.05$, noise $\delta = 0.03$, Number of directions $N = 20$, Top-performing directions $b = 4$, episode length of 500 and 600 epochs. Null policy (zero matrices) are initialized and iteratively improved in batches, reducing the computational time required during rollouts.

	Bekker Subroutine	Distance Travelled (m)	Power Consumed (kW)	Slip (%)
Earth	No	4.991	12.996	56.30
	Yes	2.181	15.765	
Mars	No	1.776	3.292	15.48
	Yes	1.501	3.444	
Moon	No	1.762	2.959	21.66
	Yes	1.38	3.098	

(a) Performance of the open-loop controller in different gravity conditions on flat ground for a simulation time of 15 s

	Bekker Subroutine	Distance Travelled (m)	Power Consumed (kW)	Slip (%)
Earth	No	5.711	13.867	35.56
	Yes	3.847	15.765	
Mars	No	1.77	3.229	15.19
	Yes	1.501	3.267	
Moon	No	1.842	2.981	17.48
	Yes	1.520	3.151	

(b) Performance of the linear policy controller in different gravity conditions on flat ground for a simulation time of 15 s

TABLE I

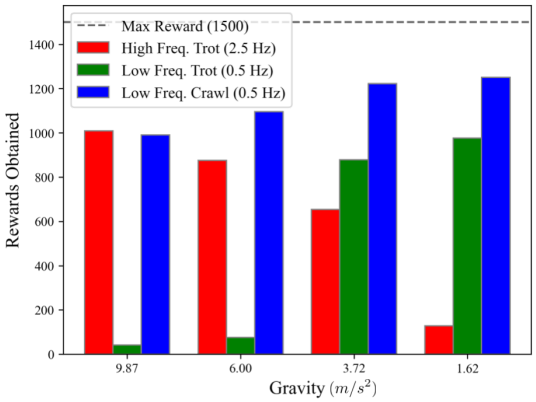

Fig. 4: Comparison of different gaits in different gravity conditions for a simulation time of 15 s

V. Results

This section contains the simulation results, a comparison to the open-loop walking controller, and the hardware tests carried out. As stated earlier, we use Nvidia Isaac Gym with the subroutine mentioned in III-C to validate our framework. We made a custom gym environment to train and evaluate the linear policies.

Each policy update took on average 12 s, and thus the total training time was about 1 h 30 min. We have learned different policies for different gravity conditions, and the walking controller was commanded to track a velocity of 0.5 m/s for Earth, 0.15 m/s for Mars and Moon. The control loop frequency for the trajectory generation and motor control was 100 Hz, and the simulation timestep was 0.01 s.

As mentioned earlier, the Bekker model subroutine introduces a slip in the total distance traveled. Despite the slip, the policy can keep the robot stable due to the shear forces occurring at the contact foot. Tables Ia and Ib show the average distance traveled and the slip occurring, which should be accounted for by the high-level path planners. As one can see, the linear policy controller outperforms the open-loop controller. The slip is calculated according to the relation $s_l = 100|(d_o - d_b)|/d_o$, where d_b and d_o is the distance travelled with and without the Bekker subroutine respectively. It is also evident that power consumption is higher as higher torques are required to produce the same motion whenever the robot traverses on loose terrains.

Fig. 4 shows the comparison of the performance of various gaits in different gravity conditions for a simulation time of 15 s. We prefer to use trot gait in higher gravity conditions to track higher velocity commands. In comparison, gaits like crawl perform better in lower gravity conditions. Such performance difference is because higher frequency gaits tend to generate higher impact forces when the leg reaches the touchdown phase. This is evident mainly due to position control methods employed. We plan to use force control methods in future work since it is possible to lower the impact forces during the touchdown or develop stable bounding behaviors irrespective of the underlying terrain type.

We have also performed preliminary hardware experiments on our proposed controller. Fig. 5 shows the keyframes of the robot traversing slope of 9° along with loose terrain made of artificial dry sand. The robot can keep itself stable when transitioning between different terrains without toppling over.

VI. Conclusion

This work successfully showed the development of a linear policy-based walking controller capable of generating robust quadrupedal motion on planetary terrains such as in flat, sloped, rugged, loose, and low-than-Earth gravity terrains. The end-foot trajectory modulating policy has been demonstrated to be transferrable across all terrain transitions. The proposed technique will provide a single framework for rapidly constructing linear feedback control policies for any multi-legged robot, thereby significantly simplifying the controller design and deployment process for planetary exploration missions.

Even though nonlinear policy parameterizations might result in better accuracy, they will have significantly higher computational requirements. Hence linear policies were considered in our framework as they have the smallest number of parameters and show good performance. The simulation results showed that our method outperforms the baseline open-loop controller by reducing slippage and offering high stability. We also observed that gaits like crawl become more stable with our framework in low gravity conditions. Preliminary results on our hardware platform Stochlite are shown to validate our framework. Future work will include deploying the robot on Lunar and Martian testbeds, performing sinkage analysis, extending the framework to force control methods, and studying more diverse gaits such as bounding. Video results can be found at: https://youtu.be/La3y-xhWm1U

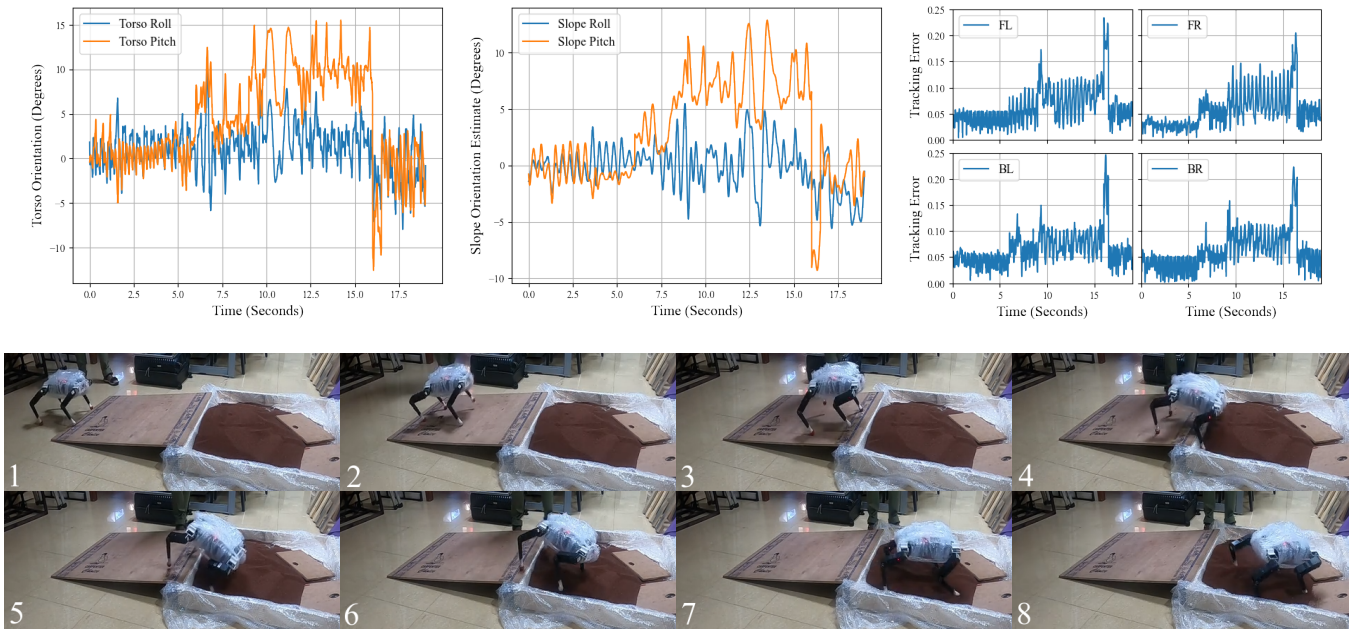

Fig. 5: Shown at the top are the plots of variation in Torso Orientation, Slope Orientation, and Tracking Errors of foot trajectory. And at the bottom keyframes of the hardware experiment performed on slope of 9° and loose terrain made of artificial sand are shown

REFERENCES

[1] D. Rodríguez-Martínez, M. van Winnendael, and K. Yoshida, "High-speed mobility on planetary surfaces: A technical review," *J. Field Robotics*, vol. 36, pp. 1436–1455, 2019.

[2] H. Kolvenbach, M. Breitenstein, C. Gehring, and M. Hutter, "Scalability analysis of legged robots for space exploration," vol. 16, pp. 10399 – 10413, Curran, 2018-06.

[3] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2245–2252, 2018.

[4] M. Hutter, C. Gehring, A. Lauber, F. Günther, D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Blösch, H. Kolvenbach, M. Bjelonic, L. Isler, and K. Meyer, "Anymal - toward legged robots for harsh environments," *Advanced Robotics*, vol. 31, pp. 918 – 931, 2017.

[5] M. H. Raibert, *Legged Robots That Balance*. USA: Massachusetts Institute of Technology, 1986.

[6] K. Byl, A. C. Shkolnik, S. Prentice, N. Roy, and R. Tedrake, "Reliable dynamic motions for a stiff quadruped," in *ISER*, 2008.

[7] 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)*, pp. 1–9, 2018.

[8] Z. Xie, X. Da, B. Babich, A. Garg, and M. van de Panne, "Glide: Generalizable quadrupedal locomotion in diverse environments with a centroidal model," 2021.

[9] 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.

[10] B. Wilcox, T. Litwin, J. Biesiadecki, J. Matthews, M. Heverly, J. Morrison, J. Townsend, N. Ahmad, A. Sirota, and B. Cooper, "Athlete: A cargo handling and manipulation robot for the moon," *Journal of Field Robotics*, vol. 24, pp. 421–434, 05 2007.

[11] S. Dirk and K. Frank, "The bio-inspired scorpion robot: Design, control & lessons learned," 2007.

[12] S. Bartsch, T. Birnschein, F. Cordes, D. Kuehn, P. Kampmann, J. Hilljegerdes, S. Planthaber, M. Roemmermann, and F. Kirchner, "Spaceclimber: Development of a six-legged climbing robot for space exploration," in *ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics)*, pp. 1–8, 2010.

[13] P. Arm, R. Zenkl, P. Barton, L. Beglinger, A. Dietsche, L. Ferrazzini, E. Hampp, J. Hinder, C. Huber, D. Schaufelberger, F. Schmitt, B. Sun, B. Stolz, H. Kolvenbach, and M. Hutter, "Spacebok: A dynamic legged robot for space exploration," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6288–6294, 2019.

[14] N. Rudin, H. Kolvenbach, V. Tsounis, and M. Hutter, "Cat-like jumping and landing of legged robots in low gravity using deep reinforcement learning," *IEEE Transactions on Robotics*, vol. 38, p. 317–328, Feb 2022.

[15] 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," 2021.

[16] K. Paigwar, L. Krishna, S. Tirumala, N. Khetan, A. Sagi, A. Joglekar, S. Bhatnagar, A. Ghosal, B. Amrutur, and S. Kolathaya, "Robust quadrupedal locomotion on sloped terrains: A linear policy approach," 2020.

[17] M. Rahme, I. Abraham, M. L. Elwin, and T. D. Murphey, "Dynamics and domain randomized gait modulation with bezier curves for sim-to-real legged locomotion," 2020.

[18] 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.

[19] M. G. Bekker, *Introduction to terrain-vehicle systems*. University of Michigan Press, 1969.

[20] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. A. Fleischmann, M. Taylor, H. Sugiyama, and D. Negrut, "Chrono: An open source multi-physics dynamics engine," in *HPCSE*, 2015.

[21] F. Zhou, R. Arvidson, K. Bennett, K. Iagnemma, C. Senatore, R. Lindemann, B. Trease, P. Bellutta, and S. Maxwell, "Simulating mars exploration rover opportunity drives using artemis," in *44th Annual Lunar and Planetary Science Conference*, 2013.

[22] H. Mania, A. Guy, and B. Recht, "Simple random search provides a competitive approach to reinforcement learning," 2018.

[23] H. Shibly, K. Iagnemma, and S. Dubowsky, "An equivalent soil mechanics formulation for rigid wheels in deformable terrain, with application to planetary exploration rovers," *Journal of Terramechanics*, vol. 42, no. 1, pp. 1–13, 2005.