

Terrain-Aware Velocity Command Smoothing for Quadruped Navigation over Unstructured Ground

Tomas Keller • Priya Raghunathan • Hana Sato

Advanced Robotics Group, Toyota Research Institute, Palo Alto, CA, USA

PUBLISHED

July 16, 2026

REGISTERED

July 18, 2026

RECORD TYPE

Preprint

ABSTRACT

Local planners for legged robots emit velocity commands without regard for what the locomotion controller can smoothly realize. On unstructured ground this mismatch produces command chatter, degrades tracking, wastes energy, and triggers premature obstacle reactions. This paper presents a terrain-aware, differentiable command-smoothing layer placed between a configuration-driven local planner in the AgniNav style and an opaque quadruped velocity interface. The layer solves a short-horizon smoothing problem whose weights are scheduled by an online terrain-difficulty estimate, and it predicts realized motion with an interface model that is initialized by GAUGE-style active calibration and updated online. Its parameters are trained end-to-end in simulation through implicit differentiation, and its prediction of the realizable motion is fed back to the planner so that collision checking follows the motion the robot is expected to execute. The approach is evaluated in an Isaac Lab deformable-terrain suite with five terrain classes and in field trials with a Unitree B2 over gravel, exposed roots, and graded slopes at 1.0 to 2.5 m/s. Compared with unsmoothed planner output, it reduces command RMS from 9.8 to 9.1 cm/s and 95th-percentile tracking error from 14.7 to 14.2 cm, raises success rate from 86.0 to 91.5 percent, lowers an energy proxy from 1519 to 1385 J/km, and increases mean speed from 1.48 to 1.68 m/s. It also outperforms a fixed gain scheduler and a GAUGE-calibrated fixed-gain smoother. Ablations and recorded failure cases show which components matter and where the approach remains fragile.

INDEX TERMS quadruped navigation, velocity command smoothing, terrain-aware control, differentiable optimization, interface calibration, local planning, Unitree B2

KAGG NOTE Smoothing velocity commands helps a legged robot only when the smoother models what the locomotion interface can realize on the current terrain and tells the planner what will actually be executed.

01 Introduction

Legged robots are increasingly deployed on ground that wheeled platforms cannot handle: loose gravel, exposed tree roots, sand, and graded slopes. Learned locomotion controllers have made such terrain routinely traversable [1, 2, 3, 4], and in most deployed systems they are exposed to the navigation stack through a narrow velocity interface: the planner sends a desired planar body twist, and the controller decides how to place the feet. This separation is convenient, because the planner can be developed, configured, and transferred across platforms without access to the controller internals [5, 6]. It also hides a mismatch. A local planner chooses commands according to geometric clearance and goal progress, but has no model of which command sequences the controller can realize smoothly on the current ground. On rough terrain the realized twist lags, saturates, and drifts relative to the command, and these effects change from one surface to the next.

The consequences are visible at the level of the whole system. When the realized motion departs from the planner’s internal prediction, the planner observes tracking deviation, replans, and issues a correction; the correction itself is imperfectly realized, and the loop can settle into command chatter. Chatter degrades tracking, increases the mechanical effort spent accelerating and decelerating the body, and causes premature obstacle reactions when the planner’s collision check is evaluated

along a path the robot will not actually follow. A standard remedy is to low-pass filter or rate-limit the commands with a fixed gain, or to schedule the gain on commanded speed. Such filters ignore the terrain, and they introduce lag that the planner is not told about. As the experiments in Section 5 show, a fixed gain scheduler can lower task success relative to sending planner output directly.

This paper addresses the gap between a configuration-driven local planner and an opaque velocity interface with a terrain-aware, differentiable command-smoothing layer. The layer solves a short-horizon smoothing problem at the control rate. It predicts realized motion through a calibrated interface model that is initialized by planner-conditioned active calibration in the sense of GAUGE [7] and refined online. Its smoothing weights are scheduled by an online estimate of terrain difficulty derived from the same interface model, and it returns its prediction of the realizable motion to the planner, so that the planner evaluates candidate commands along the trajectory the robot is expected to execute. The planner is an AgniNav-style local planner [5], which keeps the embodiment description in an explicit configuration; the smoothing layer extends that configuration with a terrain-dependent description of the velocity interface.

The contributions of this paper are as follows.

1. A formulation of velocity-command smoothing as a differentiable receding-horizon optimization layer placed

between a local planner and an opaque locomotion interface, whose weights are trained end-to-end in simulation through implicit differentiation.

2. A terrain-difficulty estimator that derives its features from the innovations and parameter drift of a GAUGE-initialized interface model updated by recursive least squares, and uses them to schedule smoothing gains online without a terrain classifier.
3. A planner feedback term through which an AgniNav-style planner scores and collision-checks candidate commands along the motion predicted by the smoothing layer instead of along ideal kinematic rollouts.
4. An evaluation in an Isaac Lab deformable-terrain suite and in field trials with a Unitree B2 over gravel, exposed roots, and graded slopes, comparing against unsmoothed planner output, a fixed gain scheduler, and a GAUGE-calibrated fixed-gain smoother, with an ablation study and an analysis of recorded failure cases.

02 Related Work

2.1 Learned locomotion behind a velocity interface

Reinforcement learning has produced quadruped controllers that track commanded body velocities over a wide range of terrain. Hwangbo et al. trained policies in simulation with a learned actuator model and transferred them to ANYmal [1]; Lee et al. extended this to challenging natural terrain with a teacher-student scheme and a terrain curriculum [2]; and Miki et al. fused exteroception with proprioception for robust perceptive locomotion in the wild [3]. RMA adapts online to unknown dynamics through an adaptation module that estimates environment extrinsics from the history of states and actions [4]. Massively parallel simulation reduced training times to minutes [8], concurrent state estimation improved robustness to estimation error [9], and controllers have been trained for high-speed running [10], for a family of gait behaviors selectable at deployment [11], and for energy-efficient gait selection [12, 13]. Perceptive policies now cross stairs, gaps, and parkour obstacles [14, 15, 16], and model-based controllers combine whole-body impulse control and model predictive control for highly dynamic motion [17, 18].

All of these controllers are commanded through a reference twist, and none of them guarantees that an arbitrary command sequence is tracked uniformly across terrain. RMA-style adaptation compensates for changes in dynamics inside the controller, but it does not inform the planner of the residual lag or gain loss that remains. The present work treats the controller as fixed and opaque, and it acts only on the command stream and on the information passed back to the planner. It is therefore complementary to controller-level adaptation and could be combined with any of the controllers above.

2.2 Local planning and navigation for legged robots

Legged navigation stacks range from end-to-end policies to modular planners. Rudin et al. trained locomotion and local navigation jointly [19], and Hoeller et al. learned a state representation and a navigation policy for cluttered, dynamic scenes [20]. Truong et al. learned robot embeddings so that navigation skills transfer across legged platforms with different dynamics [21]. Traversability-aware planners estimate where the robot can walk from images or geometry [22, 23],

and self-supervised systems such as BADGR learn navigation affordances directly from experience [24]. General navigation models trained on data from many robots provide reusable priors for goal-reaching behavior [25], and a survey by Xiao et al. catalogs learning-based planning and control for mobile robot navigation [6]. Sampling-based model predictive controllers such as MPPI [26] are widely used as local planners in off-road settings.

AgniNav [5] is the planner family used here. It is a configuration-driven, cross-embodiment local planner: the robot is described by an abstract configuration that contains the body collision envelope (front length, rear length, half width, and collision-relevant height), the sensor arrangement, and the velocity limits, and a single planning stack evaluates candidate motions against obstacles for whichever embodiment the configuration describes. Its main strength for the present problem is that embodiment-specific information is declared explicitly rather than learned implicitly, so the planner can run on the Unitree B2 without retraining. The limitation this paper targets is that the configuration describes geometry and nominal velocity limits but not the dynamic response of the velocity interface, which on legged platforms depends strongly on terrain. AgniNav rolls out candidate commands with an idealized kinematic model; on loose or uneven ground the realized motion deviates from those rollouts, and the planner reacts to the deviation by replanning. This work keeps the AgniNav planning structure and configuration interface unchanged, and adds two elements: a smoothing layer between the planner and the controller, and a feedback term that replaces the ideal rollout with the smoothing layer's prediction of the realized motion.

2.3 Identification and calibration of the command interface

Several lines of work model the relation between commanded and realized motion. Inverse kinodynamics methods learn a correction that maps desired motion to the command that achieves it on unstructured terrain [27], and VI-IKD conditions this correction on visual and inertial observations to anticipate the terrain ahead [28]. These methods correct individual commands but do not constrain how the command sequence evolves over time, so they can amplify high-frequency content when the learned inverse is steep.

GAUGE [7] is the calibration method on which this work builds. GAUGE treats the quadruped velocity controller as an opaque interface. It issues planner-conditioned probe commands, meaning probes drawn from the distribution of commands the downstream planner actually produces rather than generic excitation signals, and it fits a calibrated model of the command-to-motion response, including per-axis gain, lag, and offset, online during operation. The planner conditioning matters because a quadruped controller's response is nonlinear in the command, and a model fitted on irrelevant probes is inaccurate where it is needed. GAUGE produces an interface model but does not prescribe how that model should shape the command stream. The present work uses a GAUGE calibration as the prior for an online recursive estimate, uses the drift of that estimate as a terrain-difficulty signal, and uses the model inside a smoothing optimization. The GAUGE-calibrated fixed-gain baseline in Section 5 isolates the value of calibration alone, without online update or terrain-adaptive gains.

2.4 Differentiable optimization layers

OptNet showed that the solution of a quadratic program can be embedded as a layer in a neural network and differentiated through its optimality conditions [29]. Differentiable MPC applied the same idea to model predictive control, allowing cost and dynamics parameters to be learned from imitation or task losses [30], and differentiable convex optimization layers generalized it to disciplined convex programs [31]. The smoothing layer here is a small, strictly convex, per-axis problem whose weights are outputs of a learned function of terrain features. It uses the implicit-function-theorem gradient in the same way as these works, but with a smooth barrier formulation chosen so that a fixed number of Newton steps suffices at a 20 ms control period on embedded hardware.

2.5 Safety filters

Control barrier functions provide a minimally invasive filter that keeps a system inside a safe set [32], model predictive safety certification modifies a learning-based input only when needed to preserve recoverability [33], and safe model-based reinforcement learning uses Lyapunov arguments to restrict exploration [34]. The smoothing layer is not a safety filter: it shapes commands for tracking and efficiency, and a separate, simpler safety layer enforces the speed cap and acceleration limits. The separation is deliberate, because the smoothing weights are learned and the safety layer should not depend on learned parameters.

2.6 Simulation and sim-to-real transfer

GPU-parallel simulators such as Isaac Gym [35] and the Orbit framework [36] that evolved into Isaac Lab make it practical to train and evaluate over thousands of terrain instances. Domain randomization [37] and dynamics randomization [38, 39] remain the standard tools for transferring policies trained in simulation, and Kadian et al. showed that simulation rankings can fail to predict real-world rankings when the simulator permits behavior that does not transfer [40]. For this reason the smoothing layer is trained in simulation but deployed with an interface model that is recalibrated on the physical robot, and the evaluation includes field trials on the physical platform.

03 Problem Formulation

Time is discretized with control period Δt and indexed by k . The body twist of the robot is $y_k = (v_k^x, v_k^y, \omega_k) \in \mathbb{R}^3$, expressed in the yaw-aligned body frame, where v^x and v^y are the forward and lateral velocities and ω is the yaw rate. At each step the local planner emits a reference twist sequence $r_{k:k+H} = (r_k, \dots, r_{k+H-1})$ over a horizon of H steps, computed from the robot pose, the local obstacle representation, the goal, and the robot configuration $\mathcal{C}$. Following AgniNav [5], $\mathcal{C}$ contains the collision envelope, the sensor arrangement, and the velocity limits $\bar{v}$ and $\bar{\omega}$.

The locomotion controller is an opaque map from the commanded twist $u_k \in \mathbb{R}^3$ to the realized twist. It accepts only u_k and reports only the estimated state. Its response depends on an unobserved terrain condition ξ_k ,

$$y_{k+1} = f(y_k, u_{k-d}, \dots, u_k; \xi_k) + \epsilon_k, \quad (1)$$

where d is an input delay and ϵ_k is a disturbance due to contact events and estimation noise. The function f is unknown, and ξ_k changes as the robot moves across surfaces.

Table 1: Design constants of the smoothing layer. These are fixed settings, not tuned per experiment.

Quantity	Setting
Control period Δt	20 ms (50 Hz)
Smoothing horizon H	25 steps (0.5 s)
RLS forgetting factor λ	0.98 (memory about 1 s)
Newton iterations per step	4
Difficulty filter coefficient α	0.1
Tracking-error window T_w	1 s

A command-smoothing layer is a map $\mathcal{S}$ from the planner reference, the measured state history, and the past commands to the command sent to the controller:

$$u_k = \mathcal{S}(r_{k:k+H}, y_{0:k}, u_{0:k-1}; \eta), \quad (2)$$

with trainable parameters η . In addition, $\mathcal{S}$ may return a prediction $\hat{y}_{k+1:k+H+1}$ of the realized twist to the planner. The design objective is to choose $\mathcal{S}$ so that, over a distribution of terrain and navigation tasks, the robot (i) completes the task without collision, fall, or intervention; (ii) follows the reference path closely; (iii) issues commands with little high-frequency variation; and (iv) spends little mechanical effort on accelerating and decelerating the body. These objectives conflict. Heavy smoothing suppresses chatter but delays the response to the planner, which can increase tracking error and turn a timely avoidance maneuver into a late one. The appropriate trade-off depends on ξ_k , which is not observed directly.

04 Method

4.1 Overview

At each control step the layer (1) updates an interface model from the most recent measured transition, (2) computes a terrain-difficulty score $\varsigma_k \in [0, 1]$ from that model, (3) solves a per-axis smoothing problem whose weights are functions of ς_k , (4) passes the first command through a fixed safety layer to the controller, and (5) returns the affine map from reference to predicted realized motion to the planner, which uses it when scoring candidates in the next planning cycle. The layer runs at $\Delta t = 20$ ms. Table 1 lists the design constants.

4.2 Calibrated interface model

The layer approximates Eq. (1) with a decoupled first-order model with delay for each axis $j \in \{x, y, \psi\}$:

$$y_{k+1}^j = a_j y_k^j + b_j u_{k-d_j}^j + c_j + e_k^j, \quad (3)$$

where $a_j \in (0, 1)$ sets the response speed, $b_j > 0$ the input gain, c_j a velocity offset such as downhill drift on a slope, d_j an integer input delay, and e_k^j the one-step prediction error. The steady-state gain and time constant of the axis are

$$g_j = \frac{b_j}{1 - a_j}, \quad \tau_j = -\frac{\Delta t}{\ln a_j}. \quad (4)$$

On firm, level ground a well-trained controller has g_j close to one and a short τ_j . On gravel, sand, or roots the gain decreases because of foot slip, the time constant increases, and the error variance grows. The first-order structure is chosen because it has three parameters per axis, which can be estimated online from about one second of data, and because it keeps the smoothing problem convex. The cross-axis coupling of a real quadraped is absorbed into e_k^j ; the decoupled form also allows each axis to be solved independently.

GAUGE prior. Before each session the robot runs a GAUGE calibration [7]. GAUGE issues probe commands drawn from the planner's command distribution and returns, for each axis, a parameter estimate $\bar{\theta}_j = (\bar{a}_j, \bar{b}_j, \bar{c}_j)$, a covariance $\bar{P}_j$, and the delay d_j . These values describe the interface on the calibration surface; the delay is kept fixed, and the remaining parameters are updated online.

Online update. With $\theta_j = (a_j, b_j, c_j)^\top$ and regressor $h_k^j = (y_k^j, u_{k-d_j}^j, 1)^\top$, the estimate is updated by recursive least squares with forgetting factor λ :

$$\begin{aligned} e_k^j &= y_{k+1}^j - (h_k^j)^\top \hat{\theta}_k^j, \\ K_k^j &= \frac{P_k^j h_k^j}{\lambda + (h_k^j)^\top P_k^j h_k^j}, \\ \hat{\theta}_{k+1}^j &= \Pi_\Theta [\hat{\theta}_k^j + K_k^j e_k^j + \gamma(\bar{\theta}_j - \hat{\theta}_k^j)], \\ P_{k+1}^j &= \lambda^{-1} (P_k^j - K_k^j (h_k^j)^\top P_k^j). \end{aligned} \quad (5)$$

The small pull γ toward the GAUGE prior prevents drift when the command is nearly constant and the regressor carries little excitation, and Π_Θ projects onto the admissible set $\Theta = \{a \in [a_{\min}, a_{\max}], b \geq b_{\min}\}$, which keeps g_j and τ_j finite. The trace of P_k^j is clipped to prevent covariance windup. With $\lambda = 0.98$ the effective memory is $1/(1 - \lambda) = 50$ steps, or one second; this sets how quickly the model follows a change of surface and is the origin of the lag discussed in Section 5.5.

4.3 Terrain-difficulty estimate

Rather than classifying terrain, the layer summarizes how far the current interface departs from its calibrated behavior. For each step it forms the feature vector

$$\phi_k = \left(\bar{\nu}_k, [1 - \frac{\hat{g}_k}{\bar{g}}]_+, [\frac{\hat{\tau}_k}{\bar{\tau}} - 1]_+, \frac{|\vartheta_k|}{\vartheta_{\max}}, \bar{\sigma}_k^z \right), \quad (6)$$

where $\bar{\nu}_k$ is an exponential moving average of the normalized innovation $\nu_k = (e_k)^2 / ((h_k)^\top P_k h_k + \sigma_e^2)$ of the forward axis, $\hat{g}_k$ and $\hat{\tau}_k$ are the current forward-axis gain and time constant from Eq. (4) compared with their GAUGE values $\bar{g}$ and $\bar{\tau}$, ϑ_k is the body pitch with normalization $\vartheta_{\max}$, $\bar{\sigma}_k^z$ is the moving standard deviation of vertical body acceleration, and $[\cdot]_+ = \max(\cdot, 0)$. The first three features come from the interface model and capture slip and compliance; the last two are inexpensive proprioceptive cues for slope and roughness. The difficulty score is

$$\varsigma_k = (1 - \alpha) \varsigma_{k-1} + \alpha \sigma(w^\top \phi_k + w_0), \quad (7)$$

with logistic function σ , learned weights (w, w_0) , and filter coefficient α . The filter keeps the smoothing weights from switching faster than the smoother can usefully respond.

4.4 Differentiable smoothing layer

For each axis j the layer optimizes a command sequence $U^j = (u_k^j, \dots, u_{k+H-1}^j) \in \mathbb{R}^H$. Iterating Eq. (3) with the current estimate gives a predicted response that is affine in U^j ,

$$\hat{Y}^j = \Phi^j y_k^j + \Gamma^j U^j + \Psi^j, \quad (8)$$

where $\hat{Y}^j$ stacks $\hat{y}_{k+1}^j, \dots, \hat{y}_{k+H}^j$, Φ^j and Γ^j follow from $\hat{a}_j$ and $\hat{b}_j$, and Ψ^j collects the offset $\hat{c}_j$ and the effect of the d_j commands that have already been issued but have not yet acted.

Γ^j is lower triangular with its first d_j rows equal to zero. The smoothing cost is

$$\begin{aligned} J^j(U^j) &= \|\hat{Y}^j - R^j\|_{Q_j}^2 + \rho_1^j(\varsigma_k) \|D_1 U^j - \delta_1\|^2 \\ &\quad + \rho_2^j(\varsigma_k) \|D_2 U^j - \delta_2\|^2 \\ &\quad + \mu \sum_{i=0}^{H-1} [B(u_{k+i}^j; \bar{u}_j) + B(\Delta u_{k+i}^j; \overline{\Delta u}_j)], \end{aligned} \quad (9)$$

where R^j is the planner reference for axis j , Q_j a diagonal tracking weight, D_1 and D_2 first- and second-difference matrices, and δ_1 and δ_2 vectors that carry the previously applied commands u_{k-1}^j and u_{k-2}^j so that the differences are continuous across control steps. The smoothing weights are

$$\rho_m^j(\varsigma) = \exp(\kappa_m^j + \beta_m^j \varsigma), \quad m \in \{1, 2\}, \quad (10)$$

which keeps them positive and lets the trained parameters β_m^j determine how strongly smoothing increases with difficulty. Bounds on the command $\bar{u}_j$, taken from the AgniNav configuration, and on the command increment $\overline{\Delta u}_j$ enter through the smooth convex penalty

$$B(z; \bar{z}) = \frac{1}{s^2} [\text{sp}(s(z - \bar{z}))^2 + \text{sp}(s(-z - \bar{z}))^2], \quad (11)$$

where $\text{sp}(x) = \ln(1 + e^x)$ is the softplus function and s sets its sharpness. The penalty is nearly zero inside the bounds and grows quadratically outside them. Hard enforcement is left to the safety layer of Section 4.6, so the smoothing problem stays unconstrained, smooth, and strictly convex: D_1 is lower bidiagonal with unit diagonal and therefore full rank, so the Hessian is positive definite whenever $\rho_1^j > 0$.

Solution. The problem has $H = 25$ variables per axis. It is solved by Newton's method from the previous solution shifted by one step, with four iterations and a 25×25 Cholesky factorization per iteration. Only the first element u_k^j is applied, as in receding-horizon control.

Gradients. The trainable parameters (w, w_0, κ, β) are collected in η . The optimum U^{j*} satisfies $\nabla_U J^j = 0$, and the implicit function theorem gives

$$\frac{\partial U^{j*}}{\partial \eta} = -[\nabla_{UU}^2 J^j]^{-1} \nabla_{U\eta}^2 J^j \Big|_{U^{j*}}, \quad (12)$$

which reuses the Cholesky factor of the final Newton step. This is the same construction as OptNet and differentiable MPC [29, 30]. In contrast to unrolling the Newton iterations, it gives exact gradients at the optimum at negligible extra cost.

4.5 Planner feedback

Without feedback, the planner scores a candidate reference r^c by rolling it out through an ideal kinematic model and checking the resulting path against obstacles with the configured envelope. The smoothing layer then changes the command, and the controller changes it again. The planner therefore checks one path while the robot follows another. When the robot deviates, the planner replans and may initiate an avoidance maneuver that the actual trajectory did not require.

The feedback term closes this gap at low cost. Around the current solution the optimal smoothed sequence is approximately affine in the reference, $U^{j*} \approx M^j R^j + m^j$, where

$M^j = [\nabla_{UU}^2 J^j]^{-1} (\Gamma^j)^\top Q_j$ is available from the same factorization. The layer passes (M^j, m^j) and the prediction matrices of Eq. (8) to the planner once per planning cycle. For each candidate the planner then computes the predicted realized twist

$$\hat{Y}^{c,j} = \Phi^j y_k^j + \Gamma^j (M^j R^{c,j} + m^j) + \Psi^j, \quad (13)$$

integrates it to a predicted path, and runs the AgniNav collision check with the configured envelope on that path. It also adds the consistency cost

$$J_{fb}(r^c) = \kappa_{fb}(\varsigma_k) \sum_j \|\hat{Y}^{c,j} - R^{c,j}\|^2, \quad (14)$$

which penalizes candidates that the interface is predicted not to realize on the current terrain. The weight κ_{fb} increases with ς_k . Finally, the planner's replanning trigger compares the measured pose with the predicted path rather than with the ideal one, so deviations that the model already anticipates do not trigger replanning. Each candidate costs a few small matrix-vector products, and the AgniNav configuration interface is unchanged.

4.6 Safety layer

The applied command passes through a fixed, non-learned safety layer. It clips each axis to the configured limits, limits the per-step increment, and reduces the planar command when the measured planar speed exceeds the cap $\bar{v}$:

$$u_k^{\text{safe}} = \text{clip}(u_k) - k_s [\|y_k^{xy}\| - \bar{v}]_+ \frac{y_k^{xy}}{\|y_k^{xy}\|}, \quad (15)$$

with gain k_s and planar components $y_k^{xy} = (v_k^x, v_k^y)$. The layer also commands a controlled stop when the planner reports that no collision-free candidate exists. The safety layer is identical for all methods compared in Section 5.

4.7 Training

The parameters η are trained in simulation. Because the physics simulator is not differentiated, gradients flow through the identified interface model: the layer is run in closed loop in Isaac Lab, the transitions are logged, and windows of length T are replayed through Eqs. (3)–(12), with the interface parameters taken from the online estimate at each step. The loss over a window is

$$\mathcal{L}(\eta) = \sum_t \left(\|\hat{y}_t - r_t\|_W^2 + \lambda_1 \|u_t - u_{t-1}\|^2 + \lambda_2 \hat{E}_t + \lambda_3 [\|\hat{y}_t^{xy}\| - \bar{v}]_+^2 \right), \quad (16)$$

where $\hat{E}_t$ is the predicted positive kinetic-energy increment defined in Eq. (17) below. Data collection with the current parameters alternates with gradient updates, so the logged states follow the distribution induced by the current layer. Task success is not differentiable through this model; the feedback weight κ_{fb} is therefore selected by a grid search on simulated success rate with η held fixed. The terrain distribution during training is randomized over friction, compliance, and slope [37, 38]. On the physical robot the trained η is used without change; only the GAUGE prior is recalibrated.

05 Experiments

5.1 Setup and platform

All experiments use the Unitree B2 quadruped with its velocity interface treated as opaque, an AgniNav-style local planner configured with the B2 envelope and limits, and the safety layer of Section 4.6.

Simulation. The Isaac Lab deformable-terrain suite contains five terrain classes: gravel, exposed roots, graded slopes from 4 to 12 degrees, sand patches, and mixed terrain. Each class is evaluated with 5 seeds and 40 trials per setting using the Unitree B2 asset.

Field. Field trials were run on the TRI range over gravel, exposed roots, and graded slopes of 6, 9, and 12 degrees, with 12 runs per setting and the B2 commanded at 1.0 to 2.5 m/s.

5.2 Baselines, ablations, and metrics

Four methods are compared. All share the planner, the safety layer, and the controller.

- **Planner direct (no smoothing).** The planner reference is sent to the controller through the safety layer only.
- **Fixed gain scheduler.** The smoothing problem of Eq. (9) with a nominal interface model (unit gain, nominal lag) and smoothing weights scheduled on commanded speed by a hand-tuned lookup table, independent of terrain.
- **GAUGE-calibrated fixed gain.** The same smoothing problem with the interface model frozen at the GAUGE calibration and fixed smoothing weights. There is no online update and no terrain adaptation.
- **Ours.** The full layer with online interface estimation, terrain-adaptive weights, trained parameters, and planner feedback.

The two smoothing baselines act as plug-in filters and do not provide planner feedback. The ablation removes one component at a time from the full method. *W/o terrain-adaptive gain* fixes ς_k at its training mean. *W/o online interface estimate* freezes $\hat{\theta}$ at the GAUGE prior, so the difficulty features are computed against a fixed model. *W/o differentiable smoothing* replaces the optimization layer with a terrain-scheduled first-order low-pass filter and rate limiter whose schedule is hand-tuned, since no gradient path is available. *W/o planner feedback term* keeps the full smoothing layer but lets the planner use its native ideal rollout, without the consistency cost of Eq. (14).

The metrics are defined as follows. *Command RMS* is the root mean square of the step-to-step change of the planar command delivered to the controller, $(\frac{1}{N} \sum_k \|u_k^{xy} - u_{k-1}^{xy}\|^2)^{1/2}$, in cm/s; it measures chatter. *Tracking error p95* is the 95th percentile, over all time steps, of the distance between the measured base position and the position obtained by integrating the planner reference from the measured pose T_w earlier, in cm. *Success rate* is the percentage of trials that reach the goal within the time budget without collision, fall, or safety stop. *Mean speed* is path length divided by traversal time, in m/s. The *energy proxy* accumulates positive increments of planar body kinetic energy per kilometer traveled,

$$E = \frac{1}{L} \sum_k \left[\frac{1}{2} m_b \Delta \|v_k^{xy}\|^2 + \frac{1}{2} I_z \Delta \omega_k^2 \right]_+, \quad (17)$$

where L is the distance traveled, m_b and I_z are the nominal body mass and yaw inertia, and Δ denotes the change over

one control step. The proxy isolates the effort spent on speed changes induced by the command stream. It is not the total electrical energy of locomotion, which is dominated by the gait itself.

5.3 Main results

Table 2 reports the aggregate comparison over the evaluation protocol of Section 5.1.

Overall. The proposed layer is best on all five metrics. Relative to unsmoothed planner output it reduces command RMS from 9.8 to 9.1 cm/s (7.1%), reduces the 95th-percentile tracking error from 14.7 to 14.2 cm (3.4%), raises the success rate from 86.0% to 91.5% (5.5 percentage points), lowers the energy proxy from 1519 to 1385 J/km (8.8%), and increases mean speed from 1.48 to 1.68 m/s (13.5%). The improvement in tracking error is small in absolute terms, 0.5 cm at the 95th percentile. The larger effects are on success, energy proxy, and mean speed, which is consistent with the method’s main mechanism: it removes chatter-induced replanning and premature avoidance, rather than tightening tracking of any individual command.

Fixed gain scheduler. The speed-scheduled smoother does not reduce command RMS (9.8 cm/s, equal to the unsmoothed planner). Its tracking error is essentially unchanged (14.8 vs. 14.7 cm), and its success rate is the lowest in the table at 78.3%, 7.7 points below sending planner output directly. Its mean speed (1.58 m/s) is higher than that of the unsmoothed planner. One interpretation, supported by the first failure case in Section 5.5, is that smoothing weights tuned without regard to terrain introduce lag that the planner does not anticipate. On difficult ground the lag produces deviations, the planner corrects them, and the smoother redistributes rather than removes the resulting command variation. These data do not show which part of the success loss comes from lag and which from interaction with replanning.

GAUGE-calibrated fixed gain. Calibration alone produces a mixed result. The energy proxy falls to 1446 J/km, 73 J/km (4.8%) below the unsmoothed planner, but command RMS rises to 10.5 cm/s, tracking error rises to 16.0 cm, the highest values in the table, and success (84.2%) remains below the unsmoothed planner. A plausible explanation is that a model calibrated on one surface and then frozen inverts the wrong gain on another. When the actual gain is lower than calibrated, the smoother under-commands and then corrects, which increases step-to-step command change. The smoothing weights still remove the largest acceleration peaks, which would lower the energy proxy. The proposed method reaches an energy proxy 61 J/km (4.2%) below this baseline while also reducing command RMS and tracking error, which suggests that the online estimate and the terrain-adaptive weights are what make the calibrated model useful on varying ground.

5.4 Ablation

Table 3 removes one component at a time.

Every removal degrades both metrics. Replacing the differentiable smoothing layer with a hand-scheduled low-pass filter has the largest effect, raising tracking error by 2.6 cm to 16.8 cm and lowering success by 12.5 points to 79.0%. Removing terrain-adaptive gains raises tracking error by 1.7 cm and

lowers success by 9.1 points, and freezing the interface model raises tracking error by 1.6 cm and lowers success by 6.8 points. The two effects are similar in size, as expected, since the difficulty estimate is computed from the online model.

The planner feedback term behaves differently. Without it, tracking error increases by only 0.6 cm to 14.8 cm, but success drops by 10.0 points to 81.5%. This dissociation is consistent with the role of the term in Section 4.5. The smoothing layer alone keeps the robot close to the reference, but if the planner checks collisions along an ideal rollout that differs from the predicted realized path, candidate selection near obstacles is wrong often enough to cost task success. Tracking error alone would understate the value of the feedback term.

A further observation concerns the comparison with the baselines. Every ablated variant has a lower success rate than the unsmoothed planner (86.0%), and all have a higher tracking error than it (14.7 cm), although for the variant without planner feedback the margin is only 0.1 cm. Combined with the fixed-gain baselines in Table 2, this suggests that partial smoothing is not a safe intermediate design. A smoother that changes the command without an accurate, terrain-current model of the interface, or without informing the planner, can perform worse than no smoother. In these results the benefit appears only when the components are combined.

5.5 Failure cases

Three recorded failure cases illustrate the mechanisms discussed above.

Chatter under the fixed gain scheduler on roots. On the root course at 2.1 m/s, the fixed-gain baseline produced command chatter with a command RMS of 19.3 cm/s, roughly twice its aggregate value of 9.8 cm/s, and two leg-trip events. Both were self-recovered by the controller within 0.6 s. The episode is consistent with the interpretation of Table 2: the scheduler’s speed-based weights were not matched to the interface response on roots, and the resulting deviations fed a replanning loop. The trips were recovered by the locomotion controller rather than by the navigation stack, which suggests that this baseline depends on controller robustness for safety margin.

Replanning rate on a graded gravel slope. On gravel with a 4 degree grade, the unsmoothed planner replanned at 31 Hz and raised battery draw by 18% relative to the proposed method at matched speed. The replanning rate indicates that the planner’s deviation trigger was firing nearly continuously, because measured motion on loose gravel never matched the ideal rollout. With planner feedback, the trigger compares against the predicted realized path, so anticipated slip does not trigger replanning. The battery measurement supports the energy-proxy trend with an independent quantity, although it was recorded for this episode only and not across the full protocol.

Speed-cap overshoot at a slope crest. At a slope crest, the online interface estimate lagged the change of surface response by 0.3 s. During that interval the layer still compensated for an uphill gain deficit that no longer existed, so the realized speed briefly exceeded the speed cap. The safety layer of Eq. (15) brought the speed back within the cap in 0.4 s. This is a direct consequence of the forgetting factor in Eq. (5): a one-second memory filters contact noise but cannot follow

Table 2: Main comparison. Lower is better for command RMS, tracking error, and energy proxy; higher is better for success rate and mean speed.

Method	Command (cm/s)	RMS	Tracking error p95 (cm)	Success rate (%)	Energy proxy (J/km)	Mean speed (m/s)
Planner direct (no smoothing)	9.8	14.7	86.0	1519	1.48	
Fixed gain scheduler	9.8	14.8	78.3	1530	1.58	
GAUGE-calibrated fixed gain	10.5	16.0	84.2	1446	1.57	
Ours	9.1	14.2	91.5	1385	1.68	

Table 3: Ablation of the proposed layer.

Variant	Tracking error p95 (cm)	Success rate (%)
Ours (full)	14.2	91.5
w/o terrain-adaptive gain	15.9	82.4
w/o online interface estimate	15.8	84.7
w/o differentiable smoothing	16.8	79.0
w/o planner feedback term	14.8	81.5

an abrupt change in the response, and a crest is the sharpest such change on the course. The case shows why the safety layer is kept separate from the learned components. Possible mitigations include a pitch-rate feature that temporarily lowers the forgetting factor, or anticipating crests from the elevation map ahead of the robot.

06 Limitations

Several limitations qualify the results. First, Table 2 reports aggregate figures over the evaluation protocol; per-terrain results, separate simulation and field results, and confidence intervals across seeds and runs are not reported here. The size of the effect on individual terrain classes, and whether the margins over the strongest baseline exceed run-to-run variation, therefore cannot be judged from these data. Second, the field campaign covers one platform, one site, and three terrain types with 12 runs per setting. Sand patches and mixed terrain were evaluated only in simulation. Third, the energy proxy of Eq. (17) measures command-induced kinetic-energy changes rather than electrical consumption; the one battery measurement, from the second failure case, agrees with it in direction but does not validate it across conditions. Fourth, the decoupled first-order interface model ignores cross-axis coupling and nonlinear saturation, and its one-second memory produced the slope-crest overshoot. A richer model would need more data to estimate online. Fifth, the feedback weight is tuned by grid search on simulated success rather than learned, and the planner’s candidate selection is not differentiated. Finally, the method assumes that the controller accepts and tracks smooth twist commands; controllers that internally filter commands, or that switch gaits as a function of the command, may interact with the layer in ways not covered here.

07 Conclusion

This paper presented a terrain-aware, differentiable command-smoothing layer for quadruped navigation that sits between an AgniNav-style configuration-driven planner and an opaque velocity interface. The layer predicts realized motion with a GAUGE-initialized interface model that is updated online, schedules its smoothing weights from the drift and innovations of that model, and returns its prediction to the planner so that candidates are checked along the motion the robot is expected to execute. In the Isaac Lab deformable-terrain suite

and in Unitree B2 field trials, it improved on unsmoothed planner output, a fixed gain scheduler, and a GAUGE-calibrated fixed-gain smoother on all reported metrics, with the largest effects on success rate, energy proxy, and mean speed. The ablation and the fixed-gain baselines indicate that smoothing helps only when it is based on an interface model that tracks the current terrain and when the planner is informed of its effect; partial versions performed worse than no smoothing. Future work will address faster adaptation at abrupt terrain transitions, richer interface models with cross-axis coupling, and per-terrain evaluation with uncertainty estimates.

References

- [1] 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*, 2019. [arXiv:1901.08652](#).
- [2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning Quadrupedal Locomotion over Challenging Terrain. *Science Robotics*, 2020. [arXiv:2010.11251](#).
- [3] 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*, 2022. [arXiv:2201.08117](#).
- [4] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid Motor Adaptation for Legged Robots. *Robotics: Science and Systems*, 2021. [arXiv:2107.04034](#).
- [5] T. Zang, S. Cheng, H. Huang, S. Wang, and W. Zhang. AgniNav: Configuration-Driven Cross-Embodiment Local Planning for Robot Navigation. [arXiv:2606.10903](#), 2026.
- [6] X. Xiao, B. Liu, G. Warnell, and P. Stone. Motion Planning and Control for Mobile Robot Navigation Using Machine Learning: a Survey. *Autonomous Robots*, 2022. [arXiv:2011.13112](#).
- [7] T. Zang, H. Huang, S. Cheng, and W. Zhang. GAUGE: Planner-Conditioned Active Calibration of Opaque Quadruped Velocity Interfaces. [arXiv:2607.02841](#), 2026.
- [8] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning. *Conference on Robot Learning*, 2021. [arXiv:2109.11978](#).
- [9] G. Ji, J. Mun, H. Kim, and J. Hwangbo. Concurrent Training of a Control Policy and a State Estimator for Dynamic and Robust Legged Locomotion. *IEEE Robotics and Automation Letters*, 2022. [arXiv:2202.05481](#).
- [10] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid Locomotion via Reinforcement Learning. *Robotics: Science and Systems*, 2022. [arXiv:2205.02824](#).
- [11] G. B. Margolis and P. Agrawal. Walk These Ways: Tuning Robot Control for Generalization with Multiplicity of Behavior. *Conference on Robot Learning*, 2022. [arXiv:2212.03238](#).
- [12] Z. Fu, A. Kumar, J. Malik, and D. Pathak. Minimizing Energy Consumption Leads to the Emergence of Gaits in Legged Robots. *Conference on Robot Learning*, 2021. [arXiv:2111.01674](#).

- [13] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots. Fast and Efficient Locomotion via Learned Gait Transitions. Conference on Robot Learning, 2021. [arXiv:2104.04644](#).
- [14] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged Locomotion in Challenging Terrains using Egocentric Vision. Conference on Robot Learning, 2022. [arXiv:2211.07638](#).
- [15] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme Parkour with Legged Robots. IEEE International Conference on Robotics and Automation, 2024. [arXiv:2309.14341](#).
- [16] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot Parkour Learning. Conference on Robot Learning, 2023. [arXiv:2309.05665](#).
- [17] 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:1909.06586](#), 2019.
- [18] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis. RLOC: Terrain-Aware Legged Locomotion Using Reinforcement Learning and Optimal Control. IEEE Transactions on Robotics, 2022. [arXiv:2012.03094](#).
- [19] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter. Advanced Skills by Learning Locomotion and Local Navigation End-to-End. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2022. [arXiv:2209.12827](#).
- [20] D. Hoeller, L. Wellhausen, F. Farshidian, and M. Hutter. Learning a State Representation and Navigation in Cluttered and Dynamic Environments. IEEE Robotics and Automation Letters, 2021. [arXiv:2103.04351](#).
- [21] J. Truong, D. Yarats, T. Li, F. Meier, S. Chernova, D. Batra, and A. Rai. Learning Navigation Skills for Legged Robots with Learned Robot Embeddings. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2021. [arXiv:2011.12255](#).
- [22] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter. Fast Traversability Estimation for Wild Visual Navigation. Robotics: Science and Systems, 2023. [arXiv:2305.08510](#).
- [23] D. D. Fan, K. Otsu, Y. Kubo, A. Dixit, J. Burdick, and A. Aghamohammadi. STEP: Stochastic Traversability Evaluation and Planning for Risk-Aware Off-road Navigation. Robotics: Science and Systems, 2021. [arXiv:2103.02828](#).
- [24] G. Kahn, P. Abbeel, and S. Levine. BADGR: An Autonomous Self-Supervised Learning-Based Navigation System. IEEE Robotics and Automation Letters, 2021. [arXiv:2002.05700](#).
- [25] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. GNM: A General Navigation Model to Drive Any Robot. IEEE International Conference on Robotics and Automation, 2023. [arXiv:2210.03370](#).
- [26] G. Williams, A. Aldrich, and E. A. Theodorou. Model Predictive Path Integral Control using Covariance Variable Importance Sampling. [arXiv:1509.01149](#), 2015.
- [27] X. Xiao, J. Biswas, and P. Stone. Learning Inverse Kinodynamics for Accurate High-Speed Off-Road Navigation on Unstructured Terrain. IEEE Robotics and Automation Letters, 2021. [arXiv:2102.12667](#).
- [28] H. Karnan, K. S. Sikand, P. Atreya, S. Rabiee, X. Xiao, G. Wornell, P. Stone, and J. Biswas. VI-IKD: High-Speed Accurate Off-Road Navigation using Learned Visual-Inertial Inverse Kinodynamics. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2022. [arXiv:2203.15983](#).
- [29] B. Amos and J. Z. Kolter. OptNet: Differentiable Optimization as a Layer in Neural Networks. International Conference on Machine Learning, 2017. [arXiv:1703.00443](#).
- [30] B. Amos, I. D. J. Rodriguez, J. Sacks, B. Boots, and J. Z. Kolter. Differentiable MPC for End-to-end Planning and Control. Advances in Neural Information Processing Systems, 2018. [arXiv:1810.13400](#).
- [31] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter. Differentiable Convex Optimization Layers. Advances in Neural Information Processing Systems, 2019. [arXiv:1910.12430](#).
- [32] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control Barrier Functions: Theory and Applications. European Control Conference, 2019. [arXiv:1903.11199](#).
- [33] K. P. Wabersich and M. N. Zeilinger. Linear Model Predictive Safety Certification for Learning-Based Control. IEEE Conference on Decision and Control, 2018. [arXiv:1803.08552](#).
- [34] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause. Safe Model-based Reinforcement Learning with Stability Guarantees. Advances in Neural Information Processing Systems, 2017. [arXiv:1705.08551](#).
- [35] 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. NeurIPS Datasets and Benchmarks Track, 2021. [arXiv:2108.10470](#).
- [36] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments. IEEE Robotics and Automation Letters, 2023. [arXiv:2301.04195](#).
- [37] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. IEEE/RSJ International Conference on Intelligent Robots and Systems, 2017. [arXiv:1703.06907](#).
- [38] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. IEEE International Conference on Robotics and Automation, 2018. [arXiv:1710.06537](#).
- [39] 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. Robotics: Science and Systems, 2018. [arXiv:1804.10332](#).
- [40] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra. Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance? IEEE Robotics and Automation Letters, 2020. [arXiv:1912.06321](#).