

System Identification versus Learned Models for Commodity Quadruped Velocity Interfaces

Lin Zhao • Maximilian Weber • Julia Hoffmann • Omar Farouk

Perception and Control Group, Technical University of Munich, Munich, Germany

PUBLISHED

September 14, 2026

REGISTERED

September 17, 2026

RECORD TYPE

Preprint

ABSTRACT

Navigation stacks for commodity quadrupeds command the robot through a vendor velocity interface whose response to commands, including delay, lag, cross-axis coupling, and saturation, is undocumented and differs between platforms. Planners therefore need a calibrated model of this interface, but it is unclear whether classical system identification or a learned model should provide it. This paper compares, under matched probe budgets and on the same planner-conditioned probe data, structured identification of a state-delay-plus-saturation model, a Gaussian-process surrogate, and a multilayer perceptron with one million parameters. Scoring uses navigation-relevant criteria: worst-direction tracking error, extrapolation error on unseen command regimes, calibration time, and sensitivity to command noise. On a Unitree Go2 and a Unitree B2, with 60 probe sessions per platform, 12 m of probe motion per session, and 100 held-out command regimes per platform, the structured model attains 9.2% worst-direction tracking error and 14.6% extrapolation error, against 9.7% and 16.3% for the Gaussian process and 9.5% and 15.6% for the perceptron, with the shortest calibration time (44 s) and the lowest noise sensitivity (2.4 points at 3% noise). The margins are modest, but the failure cases are informative: at 3% command noise the perceptron’s extrapolation error on the B2 rises to 21%, whereas the structured model stays at 16.1%; conversely, the perceptron captures the B2’s saturation knee 9% better in-regime. The resulting decision rule is to identify the structured model by default, and adopt a learned model only when the saturation shape is the binding constraint.

INDEX TERMS quadruped robots, system identification, velocity interface calibration, Gaussian processes, learned dynamics models, legged robot navigation

KAGG NOTE Under matched probe budgets, a structured delay-plus-saturation model is a sufficient default for commodity quadruped velocity interfaces; a learned model earns its cost only where the shape of the saturation is what limits the planner.

01 Introduction

Commodity quadrupeds such as the Unitree Go2 and B2 are increasingly used as mobile bases for navigation research and for inspection in buildings and on stairs. On these platforms the navigation stack rarely commands joint torques. It instead sends body-frame velocity commands $\mathbf{u} = (v_x, v_y, \omega)$ to a vendor locomotion controller and treats that controller as a black box. The black box is far from an ideal integrator. It introduces transport delay and first-order lag, couples yaw-rate commands into lateral motion, ignores small commands inside a deadband, and saturates in a direction-dependent way that is not documented. A local planner that assumes $\dot{\mathbf{p}} = \mathbf{u}$ will therefore cut corners, overshoot goals, and misjudge clearance most severely in the directions where the interface deviates most from the command.

Planners therefore need a calibrated model of the velocity interface, and two families of tools can provide one. Classical system identification posits a structured model with a handful of physically meaningful parameters (delay, time constant, gain, saturation level) and estimates them from excitation data [13]. Learned models, including Gaussian processes [20, 19] and neural networks [24, 14], fit the command-to-motion map with far fewer structural assumptions. Recent active-calibration work, notably GAUGE [1], made the data side of this problem tractable: it shows how to issue planner-conditioned probe

commands that concentrate excitation where the planner will operate. What remains open is the model side. Given the same probe data and the same probe budget, which model class should a navigation stack fit, and under what conditions does a learned model justify its additional cost?

This paper addresses that question empirically and on navigation-relevant terms. Instead of average one-step prediction error, the comparison scores each model by the quantities that determine whether a planner can rely on it: the tracking error in the worst commanded direction, the error on command regimes not covered by the probes, the wall-clock time needed to obtain a usable model, and the sensitivity of the fit to noise in the logged commands. All three model classes are fitted to identical GAUGE-style probe logs with an identical motion budget of 12 m per session, on two quadrupeds of very different scale.

The contributions are as follows.

1. A navigation-oriented formulation of velocity-interface calibration, with metrics for worst-direction tracking error, extrapolation to unseen command regimes, calibration time, and probe-noise sensitivity (Section 3).
2. A structured state-delay-plus-saturation interface model with per-axis delay, lag, deadband, cross-axis gain, and smooth saturation, together with a multi-step identification

procedure and a residual diagnostic that flags unmodeled behavior instead of absorbing it (Section 4).

3. A matched-budget comparison of the structured model against a Gaussian-process surrogate and a one-million-parameter multilayer perceptron (MLP) fitted to the same probes, on a Unitree Go2 and a Unitree B2 in a laboratory and on a stair course, with 60 probe sessions and 100 held-out command regimes per platform (Section 5).
4. An analysis of the recorded failure cases and a decision rule that specifies when classical identification suffices and when a learned model buys enough to justify it (Sections 4.7 and 5.4).

02 Related Work

2.1 Active calibration of opaque velocity interfaces

The work closest to this study is GAUGE [1], which treats the quadruped velocity controller as an opaque interface. GAUGE issues planner-conditioned probe commands, that is, probes drawn from and weighted by the command distribution the downstream planner actually produces, and fits a calibrated command-to-motion response model online. Its central argument is that calibration effort should be spent where the planner operates rather than uniformly over the command space, and that the probing must respect a finite motion budget in a constrained workspace. The present paper adopts GAUGE's probing protocol without modification and asks a complementary question that GAUGE leaves open: once planner-conditioned probes have been collected, which model class should be fitted to them? Holding the probes and budget fixed isolates the effect of the model class, which allows the comparison to attribute differences in tracking, extrapolation, and noise sensitivity to modeling assumptions rather than to data collection.

Active learning for dynamics more broadly has been studied with guarantees for nonlinear system identification [16], with information-driven exploration for Gaussian-process dynamics [17], and with safety-constrained exploration-exploitation schemes for robots with uncertain dynamics [18]. These methods design the excitation signal for a fixed model class. GAUGE and the present study instead fix the excitation design and vary the model class, which is the decision a practitioner faces when integrating a calibration routine into an existing navigation stack.

2.2 Classical and nonlinear system identification

Classical system identification fits low-dimensional, interpretable models from input-output data; Schoukens and Ljung [13] give a user-oriented road map for the nonlinear case and stress that structured models with a small number of physically meaningful nonlinearities (static saturations, deadbands, delays) are often preferable when prior knowledge is available. Sparse regression over candidate nonlinearities (SINDy) [15] is a data-driven route to the same kind of parsimonious model. The structured model used here follows the block-oriented tradition: a static input nonlinearity (deadband and saturation) followed by linear dynamics with delay. The contribution of this paper is not a new identification algorithm. It is evidence about how such a model compares with flexible learned models on the specific metrics a navigation planner cares about.

Pillonetto et al. [14] survey deep networks for system identification and discuss when their flexibility pays off relative to classical model structures, emphasizing regularization and the risk of poor extrapolation outside the training distribution. The results reported below are consistent with that discussion in a robotics setting: the learned models extrapolate worse to unseen command regimes, and the MLP degrades most when the probe logs are noisy.

2.3 Gaussian-process models of robot dynamics

Gaussian processes [20] are the default nonparametric model when data are scarce, because they provide calibrated posterior variance and make few structural assumptions. Deisenroth et al. [19] show that GP dynamics models enable highly data-efficient policy learning; Hewing et al. [21] use a GP to model the residual of a nominal model inside a cautious model predictive controller; and Berkenkamp et al. [22] use GP confidence bounds to certify safe exploration. Scalable inference libraries such as GPyTorch [41] make exact GPs practical at the data sizes produced by short calibration sessions. The GP surrogate in this study is a full-model GP over a short command history. A residual GP on top of the structured model is a natural hybrid that this study does not evaluate; it is discussed as future work in Section 6.

2.4 Neural-network dynamics models

Neural dynamics models are attractive when the response has structure that is hard to parameterize. Nagabandi et al. [24] fit MLP dynamics for model-based control of legged and wheeled robots, and later added meta-learned online adaptation [25]; Chua et al. [23] use probabilistic ensembles to capture model uncertainty, building on deep ensembles [39]. In aerial robotics, hybrid models that combine a physical model with a learned residual have proven effective, from data-driven MPC with GP corrections [26] to the NeuroBEM aerodynamic model [27] and Neural-Fly's adaptive learned features [28]. These results show that learned components help most when the unmodeled effect is large and well covered by data. The B2 saturation knee examined in Section 5.4 is a legged-robot instance of exactly this situation. Neural networks are also known to be poorly calibrated out of distribution [40], which matters when a planner queries a model in command regimes the probes did not cover.

2.5 Learned locomotion controllers as the interface

The opacity of the velocity interface is largely a consequence of how modern quadruped controllers are built. Reinforcement-learned locomotion policies trained in simulation with actuator models and domain randomization [2, 9, 10, 12], often with adaptation modules or online system identification [11, 4], are deployed as velocity-tracking controllers [3, 5, 6]. Their velocity-tracking behavior is shaped by reward terms and command curricula [7, 8], not by an explicit reference model, so delay, saturation, and gait-dependent modes emerge implicitly and vary between firmware versions. Model-based controllers such as whole-body impulse control with MPC [29] and perceptive nonlinear MPC [30] expose more structure, but commodity platforms ship their controller in closed form. This study takes the controller as given and characterizes its interface from the outside.

2.6 Navigation with kinodynamic interface models

In navigation, the mismatch between commanded and executed motion is commonly handled either by learning it end-to-end or by learning an inverse kinodynamic correction. Xiao et al. survey learning-based motion planning and control for navigation [31] and propose learned inverse kinodynamics for high-speed off-road driving [32], extended with visual-inertial inputs in VI-IKD [33]. For legged robots, ArtPlanner [34] plans with explicit reachability on rough terrain, while learned approaches couple locomotion and local navigation end-to-end [35], learn robot embeddings for navigation skills [36], or couple vision and proprioception [37]. Cross-robot navigation models such as GNM [38] deliberately abstract the action interface across platforms, which shifts the burden of platform-specific calibration to a low-level layer. The velocity-interface model studied here is that layer: it is the component that lets a platform-agnostic planner issue commands whose executed motion it can predict.

03 Problem Formulation

3.1 Velocity interface

Let k index the planner control period Δt . The planner sends a body-frame command $\mathbf{u}_k = (u_k^x, u_k^y, u_k^\psi) \in \mathcal{U} \subset \mathbb{R}^3$ (forward velocity, lateral velocity, and yaw rate) to the vendor controller. Let $\mathbf{v}_k = (v_k^x, v_k^y, v_k^\psi)$ denote the executed body-frame twist, estimated from onboard odometry fused with an external reference where available. The interface is an unknown, possibly mode-dependent dynamical system

$$\mathbf{v}_{k+1} = F(\mathbf{v}_k, \mathbf{u}_k, \mathbf{u}_{k-1}, \dots, \mathbf{u}_{k-L}; \xi_k) + \mathbf{w}_k, \quad (1)$$

where L is a history length, ξ_k is an unobserved controller mode (for example, a gait or posture state), and $\mathbf{w}_k$ is process noise from terrain contact and estimation. A model $\hat{F}_\theta$ with parameters θ approximates F , and the planner uses its H -step rollout $\hat{\mathbf{v}}_{k+h|k}(\theta)$, $h = 1, \dots, H$, to predict the motion caused by a candidate command sequence.

3.2 Probe data and budget

A calibration session executes a probe sequence $\mathbf{u}_{0:T}$ and logs $\mathcal{D} = \{(\mathbf{u}_k, \mathbf{v}_k)\}_{k=0}^T$. The session is limited by a motion budget B on the distance traveled by the base position $\mathbf{p}_k$,

$$\sum_{k=0}^{T-1} \|\mathbf{p}_{k+1} - \mathbf{p}_k\|_2 \leq B, \quad B = 12 \text{ m}. \quad (2)$$

The budget reflects the practical constraint that calibration must fit in a cleared area of a laboratory or a landing and must finish quickly. A comparison is *matched* when all model classes receive probe logs generated by the same protocol under the same B .

3.3 Command regimes

A command regime $R_j \subset \mathcal{U}$ is a region of command space defined by a band of planar speed $\|(u^x, u^y)\|$, a band of commanded heading $\phi = \text{atan2}(u^y, u^x)$, a band of yaw rate, and the terrain on which it is executed (flat floor or stairs). The evaluation uses 100 held-out regimes per platform. A held-out regime is *seen* if it overlaps the region covered by the probes and *unseen* otherwise; the unseen regimes test extrapolation.

3.4 Navigation-relevant metrics

Let $\mathbf{e}_k = \hat{\mathbf{v}}_{k|k-H}(\theta) - \mathbf{v}_k$ be the H -step prediction error of the planar velocity. Partition evaluation samples by commanded heading into direction bins $\mathcal{K}_b$, $b = 1, \dots, N_b$. The *worst-direction tracking error* is

$$E_{\text{wd}} = 100 \cdot \max_b \frac{\sum_{k \in \mathcal{K}_b} \|\mathbf{e}_k\|_2}{\sum_{k \in \mathcal{K}_b} \|\mathbf{v}_k\|_2} [\%]. \quad (3)$$

The maximum over directions is deliberate. A planner's clearance margin must cover the direction in which the interface is least faithful, so the average error over directions understates the margin that is actually needed. The *extrapolation error* E_{ex} is the same normalized error ratio pooled over all samples from unseen regimes. The *calibration time* T_{cal} is the wall-clock time from the first probe command to a model ready for use by the planner, including probe execution and fitting. Finally, to measure *noise sensitivity*, the logged commands are corrupted before fitting as

$$\tilde{u}_k^i = u_k^i (1 + \epsilon_k^i), \quad \epsilon_k^i \sim \mathcal{N}(0, \sigma_u^2), \quad (4)$$

with $\sigma_u \in \{0, 0.01, 0.03\}$ (0%, 1%, and 3% command noise), and the sensitivity is the increase of the tracking error, in percentage points, between the noise-free fit and the fit at $\sigma_u = 0.03$, $S_3 = E_{\text{wd}}(\sigma_u=0.03) - E_{\text{wd}}(\sigma_u=0)$. Command noise models the discrepancy between what the planner logged and what reached the vendor controller (timestamp jitter, quantization, and dropped packets).

04 Method

4.1 Planner-conditioned probing

All model classes are fitted to probes generated by the GAUGE protocol [1]. Let $\pi_{\text{plan}}(\mathbf{u})$ be the empirical distribution of commands produced by the navigation planner on representative tasks. Probe primitives (steps, ramps, constant-curvature arcs, and direction reversals) are parameterized by a target command $\mathbf{u}^*$, and the next primitive is selected greedily as

$$\mathbf{u}^* = \arg \max_{\mathbf{u} \in \mathcal{U}_{\text{feas}}(\mathbf{p}_k)} \pi_{\text{plan}}(\mathbf{u}) \Delta \mathcal{I}(\mathbf{u} | \mathcal{D}) / c(\mathbf{u}), \quad (5)$$

where $\Delta \mathcal{I}$ is the expected reduction in predictive uncertainty of the current calibration model at $\mathbf{u}$, $c(\mathbf{u})$ is the budget consumed by the primitive, and $\mathcal{U}_{\text{feas}}(\mathbf{p}_k)$ is the set of commands whose primitive stays inside the cleared workspace from the current position. For the matched comparison, the probe logs are generated once per session and are then given unchanged to the model class assigned to that session, so that differences between classes cannot be traced to the probes.

4.2 Structured state-delay-plus-saturation model

The structured model treats each output axis $i \in \{x, y, \psi\}$ as a static input nonlinearity followed by a delayed first-order lag. The input is first passed through a symmetric deadband,

$$\bar{u}^i = \text{sign}(u^i) \max(|u^i| - \beta_i, 0), \quad (6)$$

then mixed by a gain matrix $\mathbf{G} \in \mathbb{R}^{3 \times 3}$ whose diagonal holds the per-axis gains and whose off-diagonal terms capture cross-coupling (most importantly, yaw rate leaking into lateral velocity during turns), and then saturated by a smooth limiter,

$$s_i(z) = \bar{s}_i \tanh(z/\bar{s}_i), \quad z^i = [\mathbf{G} \bar{\mathbf{u}}]_i. \quad (7)$$

The executed velocity then follows a first-order lag with an integer transport delay d_i ,

$$v_{k+1}^i = a_i v_k^i + (1-a_i) s_i(z_{k-d_i}^i), \quad a_i = \exp(-\Delta t/\tau_i), \quad (8)$$

with time constant $\tau_i > 0$. The parameter vector is $\theta = \{d_i, \tau_i, \beta_i, \bar{s}_i\}_i \cup \{\mathbf{G}\}$, that is, three integer delays and 18 continuous parameters. This structure encodes what is known about velocity-tracking locomotion controllers: they respond after a communication and gait-phase delay, converge with roughly exponential settling, ignore very small commands, and cap speed. The tanh limiter fixes the *shape* of the saturation knee up to its level $\bar{s}_i$. This is the main structural restriction of the model, and Section 5.4 shows where it binds.

4.3 Identification procedure

Because the planner consumes multi-step rollouts, the continuous parameters are fitted on the multi-step simulation error rather than on the one-step equation error, which is biased under input noise and does not penalize the accumulation of lag errors. For each delay triple $\mathbf{d} \in \{0, \dots, D\}^3$,

$$\hat{\theta}_{\mathbf{d}} = \arg \min_{\theta} \sum_{k \in \mathcal{K}_{\mathbf{d}}} \sum_{h=1}^H \|\mathbf{v}_{k+h} - \hat{\mathbf{v}}_{k+h|\mathbf{d}}(\theta)\|_{\Sigma}^2 + \lambda \|\theta - \theta_0\|_2^2, \quad (9)$$

where Σ is the per-axis noise covariance estimated from stationary segments of the log, θ_0 is a weak prior centered on unit gain, zero coupling, and the vendor's nominal speed limits, and λ is a small ridge weight. Problem (9) is solved by Levenberg-Marquardt with the gains initialized by linear least squares on the unsaturated samples. The delays are chosen as $\hat{\mathbf{d}} = \arg \min_{\mathbf{d}}$ of the converged objective, and the axes decouple sufficiently that the search can be done per axis. With 21 parameters, the fit completes in well under the time needed to execute the probes, which is why the calibration time of the structured model is dominated by probing.

4.4 Residual diagnostic

A structured model cannot represent everything; the goal is for its failures to be visible. After fitting, the normalized residual of each probed command region R_j is computed as

$$r_j = \frac{1}{|\mathcal{K}_j|} \sum_{k \in \mathcal{K}_j} \mathbf{e}_k^T \Sigma^{-1} \mathbf{e}_k, \quad (10)$$

and R_j is flagged if r_j exceeds the χ^2 quantile at level $1 - \alpha$ appropriate to its sample count. A flagged region is reported to the navigation stack as one where the model is not trusted, so that the planner can inflate its margins there or avoid the regime. Flagging, rather than absorbing, is what distinguishes the structured model's handling of the Go2 mode hysteresis in Section 5.4.

4.5 Gaussian-process surrogate

The GP surrogate models the velocity increment $\Delta \mathbf{v}_k = \mathbf{v}_{k+1} - \mathbf{v}_k$ as an independent GP per output axis, with input $\mathbf{x}_k = (\mathbf{v}_k, \mathbf{u}_k, \dots, \mathbf{u}_{k-L})$, so that it can represent delay and lag implicitly through the command history. Each output uses

an automatic-relevance-determination squared-exponential kernel,

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \Lambda^{-1}(\mathbf{x} - \mathbf{x}')\right), \quad (11)$$

with hyperparameters $(\sigma_f, \Lambda, \sigma_n)$ obtained by maximizing the log marginal likelihood [20]. The posterior mean at a test input $\mathbf{x}_*$ is $\mu(\mathbf{x}_*) = \mathbf{K}_*^T (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$, and multi-step predictions are obtained by propagating the mean. Inference is exact, which is feasible at the data sizes of a 12 m session [41]. Hyperparameter optimization is the dominant fitting cost and accounts for the GP's longer calibration time.

4.6 Multilayer perceptron

The MLP uses the same input $\mathbf{x}_k$ and predicts the increment, $\hat{\mathbf{v}}_{k+1} = \mathbf{v}_k + f_\phi(\mathbf{x}_k)$, where f_ϕ has approximately 10^6 parameters, organized as five fully connected hidden layers of width 512 with ELU activations. The weights are trained with Adam [42] on the multi-step loss of Eq. (9) with θ replaced by ϕ , with weight decay in place of the prior term and early stopping on a held-out portion of the session's probe log. The network is deliberately larger than the data strictly require. This is the regime in which practitioners commonly deploy off-the-shelf learned dynamics models, and it exposes the overfitting risk examined in Section 5.4.

4.7 Decision rule

The comparison motivates a rule that a navigation stack can apply automatically after a structured fit. Let $\mathcal{K}_{\text{knee}}$ be the set of samples whose saturation argument lies within the knee band $|z^i| \in [\gamma_1 \bar{s}_i, \gamma_2 \bar{s}_i]$, let ρ_{knee} be the fraction of the total normalized residual of Eq. (10) that falls in that band, and let $\pi_{\text{plan}}(\mathcal{K}_{\text{knee}})$ be the fraction of planner commands that fall in it. The rule selects the learned model if and only if all three conditions below hold, and the structured model otherwise:

$$\text{learned} \iff \begin{cases} \rho_{\text{knee}} \geq \bar{\rho}, \\ \pi_{\text{plan}}(\mathcal{K}_{\text{knee}}) \geq \eta, \\ \hat{\sigma}_u \leq \bar{\sigma}, \end{cases} \quad (12)$$

In words: the structured model is the default. A learned model is adopted only if (i) the structured model's error is concentrated at the saturation knee, (ii) the planner actually operates there, so that the saturation shape is the binding constraint on performance, and (iii) the estimated command noise $\hat{\sigma}_u$ is low enough that the learned model will not overfit it. Residuals concentrated elsewhere, for example in a hysteretic mode transition, do not trigger the learned model. They are flagged by the diagnostic of Eq. (10) and handled by the planner as untrusted regimes. The thresholds $\bar{\rho}$, η , and $\bar{\sigma}$ are design parameters; this paper derives the structure of the rule from the recorded failure cases but does not report a prospective validation of particular threshold values.

4.8 Use of the model by the planner

In deployment, the calibrated model is used in two places. The local planner rolls out $\hat{F}_\theta$ over its horizon to predict executed trajectories for collision checking. A command pre-compensator also inverts the model to find the command that best realizes the planner's reference twist $\mathbf{v}^{\text{ref}}$ after the identified delay,

$$\mathbf{u}_k = \arg \min_{\mathbf{u} \in \mathcal{U}} \left\| \hat{\mathbf{v}}_{k+\hat{d}+1|k}(\mathbf{u}) - \mathbf{v}_{k+\hat{d}+1}^{\text{ref}} \right\|_{\mathbf{W}}^2 + \mu \left\| \mathbf{u} - \mathbf{u}_{k-1} \right\|_2^2, \quad (13)$$

with weighting $\mathbf{W}$ and smoothing weight μ . For the structured model the inverse is available in closed form away from saturation, because the lag, gain, and deadband are invertible; for the GP and MLP it requires a small numerical optimization at every control step.

05 Experiments

5.1 Platforms and protocol

Experiments use a Unitree Go2 and a Unitree B2, commanded through their vendor velocity interfaces, in a laboratory and on a stair course. For each platform, 60 matched probe sessions were run with GAUGE-style planner-conditioned probes: 30 sessions whose probe logs were used for classical identification of the structured model and 30 whose logs were used for the learned models, with the GP surrogate and the MLP fitted to the same probe logs. Every session used the identical probe budget of 12 m of motion. To measure sensitivity to probe noise, each log was additionally fitted at three command-noise levels, 0%, 1%, and 3%, following Eq. (4). All models were scored on 100 held-out command regimes per platform with the metrics of Section 3. The tables report results aggregated over both platforms; platform-specific numbers are quoted where they appear in the failure analysis.

REPRODUCIBILITY NOTE

All three model classes receive identical inputs: the same planner-conditioned probe protocol, the same 12 m motion budget per session, the same three command-noise levels applied to the logged data, and the same 100 held-out command regimes per platform. The metrics are defined in Eqs. (3)–(4).

5.2 Main comparison

Table 1 reports the main comparison. The structured model has the lowest value in every column, but the margins differ considerably in size and should be read accordingly. On worst-direction tracking error the three classes lie within 0.5 percentage points of each other: 9.2% for the structured model, 9.5% for the MLP, and 9.7% for the GP. In other words, when the command regime is covered by the probes, all three model classes capture the interface about equally well. The separation is larger on extrapolation to unseen regimes: 14.6% for the structured model against 15.6% for the MLP (1.0 point higher) and 16.3% for the GP (1.7 points higher). This is the expected consequence of structure. The deadband, lag, and tanh limiter constrain how the structured model behaves outside the probed region, whereas the GP reverts toward its prior mean and the MLP's behavior there is unconstrained.

Calibration time is dominated by probe execution, which is common to all classes, so the differences are small: 44 s for the structured model, 46 s for the GP, and 48 s for the MLP, that is, 2 s and 4 s longer, respectively, attributable to fitting. Noise sensitivity is also similar in magnitude: the structured model loses 2.4 points at 3% command noise, against 2.6 for the MLP and 2.7 for the GP. Taken together, Table 1 does not show that learned models are inadequate. It shows that, under a

12 m budget, they do not buy additional accuracy on any of the four navigation-relevant criteria in aggregate. The aggregate results are reported without per-session dispersion, and the sub-point differences in tracking error should not be read as decisive on their own. The informative evidence is in the failure cases below.

5.3 Model-class ablation

Table 2 isolates the model class on the worst-direction tracking metric. The structured model again attains 9.2% and the GP surrogate 9.7%. The ablation records 9.8% for the 1M-parameter MLP, which is 0.3 points above the 9.5% listed for the MLP in Table 1. Under this record, the MLP ranks last rather than second on tracking error, and the gap between structured and learned models is 0.5 to 0.6 points. Neither table changes the qualitative conclusion: on in-regime tracking the model classes are close, and the structured model is never worse.

5.4 Failure cases

Three recorded failure cases explain where the aggregate numbers come from and motivate the decision rule of Section 4.7.

Go2 mode hysteresis. The Go2's locomotion controller exhibits an undocumented mode hysteresis: its response depends on the recent command history, not only on the current command. No static input nonlinearity in Eqs. (6)–(8) can represent this, and the structured model does not try to. Its residual exceeds the threshold of Eq. (10) in the affected regimes, which are flagged as fit residuals and passed to the planner as untrusted. The GP surrogate can represent the hysteresis through its command history input, but it fits it only when given 1.8 times the probe budget, about 21.6 m of probe motion instead of 12 m. Under the matched budget, therefore, neither model captures the hysteresis. The difference is that the structured model reports its failure explicitly, whereas the GP does not.

B2 at 3% command noise. When 3% noise is applied to the logged commands, the MLP's extrapolation error on the B2 rises to 21%, whereas the structured model stays at 16.1%, a gap of 4.9 points. The MLP has enough capacity to fit the noise in the probes, and the overfit function is unconstrained in unseen regimes. The structured model's 21 parameters leave little room to absorb the noise in the same way, which is consistent with its smaller degradation. Because the pooled noise sensitivities in Table 1 differ by only 0.2 points, this case shows that aggregate sensitivity can conceal a much larger platform-specific degradation in extrapolation.

B2 saturation knee: where the learned model wins. The B2's speed saturation has a nonlinear knee whose shape departs from the tanh limiter of Eq. (7). In the regimes around this knee, the MLP captures the response 9% better than the structured model. This is the one recorded situation in which the extra flexibility of a learned model pays off, and it is exactly the situation the decision rule of Eq. (12) is designed to detect. The rule returns *learned* only when the saturation shape is the binding constraint, that is, when the structured model's residual concentrates at the knee and the planner commands speeds in that band, and only when the probe logs are clean enough that the noise-induced failure above does not apply.

5.5 Discussion

The results suggest a division of labor rather than a winner. Delay, lag, deadband, and saturation level are well captured by a model with 21 parameters, and that model degrades predictably

Table 1: Main comparison on the Go2 and B2 under matched probe budgets (12 m per session). Lower is better in every column. Noise sensitivity is the increase in error, in percentage points, at 3% command noise.

Method	Worst-direction tracking error (%)	Extrapolation error (unseen regimes, %)	Calibration time (s)	Noise sensitivity (error delta at 3%, pts)
Gaussian-process surrogate (same probes)	9.7	16.3	46	2.7
MLP (same probes, 1M params)	9.5	15.6	48	2.6
Ours (structured state-delay + saturation)	9.2	14.6	44	2.4

Table 2: Model-class ablation on worst-direction tracking error.

Variant	Worst-direction tracking error (%)
Structured state-delay + saturation (ours, classical)	9.2
Gaussian-process surrogate	9.7
MLP 1M params	9.8

outside the probed region and under command noise. Learned models do not improve on it in aggregate under a 12 m budget, extrapolate worse, and in the MLP’s case can fail badly when the probes are noisy. Their advantage is local, at a saturation knee whose shape the structured model cannot represent, and it matters only if the planner spends time there. For a navigation stack that rarely commands near the speed limit, as is typical when stepping onto stairs, the structured model suffices. For a stack that routinely drives a B2 at its limit, the decision rule recommends the learned model, provided the probe logs are clean.

06 Limitations

Several limitations qualify these conclusions. First, the tables report aggregate values without confidence intervals or per-session dispersion, and several differences in Table 1 are below one percentage point; a statistical comparison over the 30 sessions per class and platform is needed before the in-regime tracking differences can be treated as reliable. The two tables also list different tracking errors for the MLP (9.5% and 9.8%). Second, the study covers two platforms from a single vendor, one budget (12 m), and two environments (a laboratory and a stair course). Outdoor terrain, payload changes, and firmware updates, all of which alter the interface, were not examined. Third, the learned baselines are a full-model GP and a single MLP architecture. Hybrid models, such as a residual GP or residual network on top of the structured model [21, 26], ensembles [39], and smaller or more strongly regularized networks may close the extrapolation and noise gaps. Fourth, the decision rule is derived from the recorded failure cases, and its thresholds have not been validated prospectively. Finally, the scoring is by model error, not by closed-loop navigation outcomes such as collisions or completion time. Model error is a proxy for those outcomes, and the strength of the link has not been measured here.

07 Conclusion

This paper compared classical identification of a structured state-delay-plus-saturation model with a Gaussian-process surrogate and a one-million-parameter MLP for characterizing the velocity interface of commodity quadrupeds, under matched planner-conditioned probe budgets on a Unitree Go2 and a Unitree B2. The structured model achieved the lowest worst-direction tracking error (9.2%), extrapolation error (14.6%), calibration time (44 s), and noise sensitivity (2.4 points), although the in-regime margins over the learned models are small.

The failure cases are more informative than the averages: the structured model flags behavior it cannot represent instead of absorbing it, it is robust where the MLP overfits noisy probes, and it loses only where the saturation shape itself is what limits the planner. The resulting decision rule is simple: identify the structured model by default, and adopt a learned model only when the saturation knee binds and the probes are clean.

References

- [1] T. Zang, H. Huang, S. Cheng, and W. Zhang. *GAUGE: Planner-Conditioned Active Calibration of Opaque Quadruped Velocity Interfaces*. arXiv:2607.02841, 2026.
- [2] 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, arXiv:1901.08652, 2019.
- [3] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. *Learning Quadrupedal Locomotion over Challenging Terrain*. Science Robotics, arXiv:2010.11251, 2020.
- [4] A. Kumar, Z. Fu, D. Pathak, and J. Malik. *RMA: Rapid Motor Adaptation for Legged Robots*. Robotics: Science and Systems, arXiv:2107.04034, 2021.
- [5] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. *Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning*. Conference on Robot Learning, arXiv:2109.11978, 2021.
- [6] 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, arXiv:2201.08117, 2022.
- [7] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. *Rapid Locomotion via Reinforcement Learning*. Robotics: Science and Systems, arXiv:2205.02824, 2022.
- [8] G. B. Margolis and P. Agrawal. *Walk These Ways: Tuning Robot Control for Generalization with Multiplicity of Behavior*. Conference on Robot Learning, arXiv:2212.03238, 2022.
- [9] 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, arXiv:1804.10332, 2018.
- [10] 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, arXiv:1710.06537, 2018.
- [11] W. Yu, J. Tan, C. K. Liu, and G. Turk. *Preparing for the Unknown: Learning a Universal Policy with Online System Identification*. Robotics: Science and Systems, arXiv:1702.02453, 2017.
- [12] 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, arXiv:1703.06907, 2017.
- [13] J. Schoukens and L. Ljung. *Nonlinear System Identification: A User-Oriented Road Map*. IEEE Control Systems Magazine, arXiv:1902.00683, 2019.

- [14] G. Pilonetto, A. Aravkin, D. Gedon, L. Ljung, A. H. Ribeiro, and T. B. Schön. [Deep Networks for System Identification: A Survey](#). arXiv:2301.12832, 2023.
- [15] S. L. Brunton, J. L. Proctor, and J. N. Kutz. [Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems](#). Proceedings of the National Academy of Sciences, arXiv:1509.03580, 2016.
- [16] H. Mania, M. I. Jordan, and B. Recht. [Active Learning for Nonlinear System Identification with Guarantees](#). Journal of Machine Learning Research, arXiv:2006.10277, 2022.
- [17] M. Buisson-Fenet, F. Solowjow, and S. Trimpe. [Actively Learning Gaussian Process Dynamics](#). Learning for Dynamics and Control, arXiv:1911.09946, 2020.
- [18] T. Lew, A. Sharma, J. Harrison, A. Bylard, and M. Pavone. [Safe Active Dynamics Learning and Control: A Sequential Exploration-Exploitation Framework](#). IEEE Transactions on Robotics, arXiv:2008.11700, 2022.
- [19] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. [Gaussian Processes for Data-Efficient Learning in Robotics and Control](#). IEEE Transactions on Pattern Analysis and Machine Intelligence, arXiv:1502.02860, 2015.
- [20] C. E. Rasmussen and C. K. I. Williams. [Gaussian Processes for Machine Learning](#). MIT Press, ISBN 0-262-18253-X, 2006.
- [21] L. Hewing, J. Kabzan, and M. N. Zeilinger. [Cautious Model Predictive Control Using Gaussian Process Regression](#). IEEE Transactions on Control Systems Technology, arXiv:1705.10702, 2017.
- [22] F. Berkenkamp, M. Turchetta, A. P. Schoellig, and A. Krause. [Safe Model-based Reinforcement Learning with Stability Guarantees](#). Advances in Neural Information Processing Systems, arXiv:1705.08551, 2017.
- [23] K. Chua, R. Calandra, R. McAllister, and S. Levine. [Deep Reinforcement Learning in a Handful of Trials Using Probabilistic Dynamics Models](#). Advances in Neural Information Processing Systems, arXiv:1805.12114, 2018.
- [24] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. [Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning](#). IEEE International Conference on Robotics and Automation, arXiv:1708.02596, 2018.
- [25] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn. [Learning to Adapt in Dynamic, Real-World Environments through Meta-Reinforcement Learning](#). International Conference on Learning Representations, arXiv:1803.11347, 2019.
- [26] G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza. [Data-Driven MPC for Quadrotors](#). IEEE Robotics and Automation Letters, arXiv:2102.05773, 2021.
- [27] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, and D. Scaramuzza. [NeuroBEM: Hybrid Aerodynamic Quadrotor Model](#). Robotics: Science and Systems, arXiv:2106.08015, 2021.
- [28] M. O’Connell, G. Shi, X. Shi, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung. [Neural-Fly Enables Rapid Learning for Agile Flight in Strong Winds](#). Science Robotics, arXiv:2205.06908, 2022.
- [29] 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.
- [30] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter. [Perceptive Locomotion through Nonlinear Model-Predictive Control](#). IEEE Transactions on Robotics, arXiv:2208.08373, 2023.
- [31] X. Xiao, B. Liu, G. Warnell, and P. Stone. [Motion Planning and Control for Mobile Robot Navigation Using Machine Learning: A Survey](#). Autonomous Robots, arXiv:2011.13112, 2022.
- [32] 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, arXiv:2102.12667, 2021.
- [33] H. Karnan, K. S. Sikand, P. Atreya, S. Rabiee, X. Xiao, G. Warnell, P. Stone, and J. Biswas. [VI-IKD: High-Speed Accurate Off-Road Navigation Using Learned Visual-Inertial Inverse Kinodynamics](#). IEEE International Conference on Robotics and Automation, arXiv:2203.15983, 2022.
- [34] L. Wellhausen and M. Hutter. [ArtPlanner: Robust Legged Robot Navigation in the Field](#). Field Robotics, arXiv:2303.01420, 2023.
- [35] 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, arXiv:2209.12827, 2022.
- [36] J. Truong, D. Yarats, T. Li, F. Meier, S. Chernova, D. Ba-tra, and A. Rai. [Learning Navigation Skills for Legged Robots with Learned Robot Embeddings](#). IEEE/RSJ International Conference on Intelligent Robots and Systems, arXiv:2011.12255, 2021.
- [37] Z. Fu, A. Kumar, A. Agarwal, H. Qi, J. Malik, and D. Pathak. [Coupling Vision and Proprioception for Navigation of Legged Robots](#). IEEE/CVF Conference on Computer Vision and Pattern Recognition, arXiv:2112.02094, 2022.
- [38] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. [GNM: A General Navigation Model to Drive Any Robot](#). arXiv:2210.03370, 2022.
- [39] B. Lakshminarayanan, A. Pritzel, and C. Blundell. [Simple and Scalable Predictive Uncertainty Estimation Using Deep Ensembles](#). Advances in Neural Information Processing Systems, arXiv:1612.01474, 2017.
- [40] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. [On Calibration of Modern Neural Networks](#). International Conference on Machine Learning, arXiv:1706.04599, 2017.
- [41] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson. [GPYtorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration](#). Advances in Neural Information Processing Systems, arXiv:1809.11165, 2018.
- [42] D. P. Kingma and J. Ba. [Adam: A Method for Stochastic Optimization](#). International Conference on Learning Representations, arXiv:1412.6980, 2015.