

# How Much Future Range Do You Need? An Ablation Study for Agile Quadruped Navigation

Minju Seo • Junho Kang • Hyesun Bae

Legged Locomotion and Control Lab, KAIST, Daejeon, South Korea

PUBLISHED

August 14, 2026

REGISTERED

August 16, 2026

RECORD TYPE

Preprint

## ABSTRACT

Future-range predictors estimate how far a legged robot can travel along control-aligned directions within a prediction horizon  $H$ , but the horizon is usually fixed by hand even though the appropriate look-ahead depends on speed and terrain. In a systematic ablation with the full navigation stack held fixed, the horizon (0.25, 0.5, 1.0, and 2.0 s), the query density, and the number of control-aligned directions are varied for a simulated Unitree Go2 in Isaac Lab across 12 scene families, speeds from 1 to 5 m/s, and flat, mixed, and rough terrain. From these results, a lightweight rule is fitted that selects the horizon from the commanded speed and an online terrain estimate, and the rule is then run on a physical Go2. In the simulated main comparison, the selected rule reaches a 92.3 percent success rate, compared with 81.1 percent for the best fixed horizon and 80.9 percent without future range, while maintaining a mean speed of 3.30 m/s. In the horizon sweep, success peaks at  $H = 0.5$  s and declines for shorter and longer horizons; long horizons are most harmful on occluded turns at 4 m/s, where  $H = 2.0$  s produces 4.7 braking events per km. The rule adds 0.4 ms of decision latency, and on hardware it selected horizons between 0.5 and 1.0 s in 8/8 recorded runs. The fixed-horizon entries are not consistent across the recorded tables, and in the sweep the rule matches but does not exceed the best fixed horizon.

**INDEX TERMS** quadruped navigation, future range prediction, prediction horizon, ablation study, legged locomotion, Unitree Go2

**KAGG NOTE** The useful future-range horizon is a function of speed and terrain rather than a constant: in the recorded sweep a mid-range horizon of 0.5 s is best on success, tail tracking error, and braking at once, while the longest horizon is harmful exactly where the robot cannot see, on occluded turns.

## 01 Introduction

Agile quadruped navigation couples two problems that are usually studied separately. A locomotion controller must track a commanded body velocity over uneven ground, and a local planner must choose that velocity so that the robot keeps moving toward its goal without colliding. Learned locomotion controllers now track commands robustly on rough terrain and at several meters per second [6, 4, 2, 3, 10], which shifts the bottleneck to perception and planning: at 4–5 m/s a quadruped covers its own body length in a fraction of a second, and the planner needs to know not only where obstacles are now but whether the motions it is about to command will remain free.

FutureRay [1] addresses this mismatch with a perception module that predicts *future free range* along the robot’s own predicted, control-aligned directions. Instead of measuring clearance along fixed sensor rays, it queries the space that candidate commands would sweep over a horizon  $H$ , so that the obstacle information delivered to the controller is expressed in the same coordinates as the actions the controller can execute. This alignment is attractive for legged robots, whose executed paths curve with yaw rate and deviate from straight sensor rays at speed.

A future-range predictor, however, introduces a design parameter that has received little systematic attention: the horizon  $H$ . A short horizon truncates the predicted range, so at high speed the reported free distance may fall below the stopping dis-

tance even in open space, and the planner brakes unnecessarily. A long horizon extrapolates candidate motions far beyond what the sensor currently observes, and around occluders the predictor must report range for space it cannot see. Query density (how many points are evaluated along each predicted path) and direction count (how many candidate commands are evaluated) set a further accuracy–compute trade-off. Existing systems typically fix these values once, and to our knowledge no study has varied them while holding the rest of a legged navigation stack constant.

This paper reports such a study. The navigation stack—locomotion policy, local planner, speed governor, and safety logic—is held fixed, and only the future-range predictor’s horizon, query density, and direction count are varied, over speeds from 1 to 5 m/s and three terrain classes on a simulated Unitree Go2 in Isaac Lab. From the resulting data a lightweight horizon-selection rule is fitted that adapts  $H$  online from the commanded speed and an estimated terrain class, and the rule is run on a physical Go2.

The contributions are:

1. A controlled ablation protocol for future-range predictors in which the predictor configuration  $(H, \rho, K)$  is the only independent variable, evaluated over 12 scene families, 5 seeds, 25 trials per scene–seed instance, five speeds, and three terrain classes.

2. An analysis of the horizon trade-off that links a truncation-induced lower bound on  $H$  (from stopping distance and gait-transition time) to an extrapolation-induced penalty for long horizons, and that explains the recorded failure modes on occluded turns and rough terrain.
3. A lightweight horizon-selection rule, a speed-terrain lookup with hysteresis driven by a proprioceptive terrain estimator, that adds 0.4 ms of decision latency.
4. An evaluation of the rule against fixed horizons and a present-only baseline, together with an explicit account of where the recorded evidence supports the rule and where it does not, including hardware runs on a Go2.

## 02 Related Work

### 2.1 Learned locomotion as the execution layer

Reinforcement learning in simulation is now the dominant route to robust quadruped controllers. Hwangbo et al. [6] trained ANYmal policies with a learned actuator network and showed that agile behaviors transfer to hardware, and Tan et al. [7] showed that careful system identification and randomization allow gaits learned in simulation to run on a Minitaur. Lee et al. [4] used a teacher-student scheme with privileged terrain information to obtain blind locomotion over challenging natural terrain, and Kumar et al. [5] introduced Rapid Motor Adaptation, in which an adaptation module infers environment extrinsics from proprioceptive history. Ji et al. [8] trained a state estimator concurrently with the policy to estimate body velocity and foot contact, and Kim et al. [9] replaced reward shaping for safety-relevant quantities with explicit constraints. Rudin et al. [2] showed that massively parallel simulation makes such policies trainable in minutes, and Margolis et al. [10] pushed learned velocity tracking on the Mini Cheetah to high speeds. Walk These Ways [11] exposed gait parameters as command inputs so that a single policy can produce a family of behaviors. These works establish the execution layer that our study treats as fixed: a velocity-tracking policy whose tracking quality depends on terrain and speed. We do not modify the locomotion controller; rather, we ask how much look-ahead the perception module should give a planner that commands such a controller. The terrain dependence documented in these works, and in particular the need for gait adaptation on rough ground, motivates our use of terrain as an input to the horizon rule.

### 2.2 Perceptive locomotion and legged navigation

A second line of work adds exteroception to the locomotion policy itself. Miki et al. [3] fused proprioception with a noisy elevation map through a belief encoder, allowing ANYmal to traverse hiking trails, and Agarwal et al. [12] trained a policy directly from egocentric depth. Yang et al. [41] combined proprioception and depth with cross-modal transformers to step over obstacles. Parkour systems [13, 14, 15] push this direction to highly dynamic maneuvers; ANYmal Parkour [15] in particular separates a perception module, a set of locomotion skills, and a navigation module that selects skills and targets. Grandia et al. [24] integrated terrain perception into nonlinear model predictive control, and Hoeller et al. [25] learned a neural scene representation that completes occluded terrain for locomotion. These works use perception to decide where feet go; our concern is one level above, where perception decides which body velocities remain collision-free. The occlusion comple-

tion of [25] is related to our failure analysis: a long future-range horizon effectively asks the predictor to complete unobserved space, and our results indicate that this is where long horizons hurt.

Several systems learn the coupling between locomotion and navigation directly. Rudin et al. [16] trained a single policy for locomotion and local navigation end-to-end, and Hoeller et al. [17] learned a state representation for navigation in cluttered and dynamic environments. Fu et al. [18] coupled a vision-based planner with a proprioceptive locomotion policy whose estimated velocity informs the planner, Truong et al. [19] learned navigation policies with robot embeddings for legged platforms, and ViNL [20] combined visual navigation with obstacle-aware locomotion. Kim et al. [21] learned a forward dynamics model of a quadruped and an informed trajectory sampler, so that candidate commands are evaluated by predicting the motion they produce; this is close in spirit to the control-aligned queries of FutureRay, but the prediction is of the robot's own motion rather than of free range. Lee et al. [23] learned robust navigation and locomotion for wheeled-legged robots in urban settings. Compared with end-to-end approaches, our modular stack fixes the interface between perception and planning, which is what allows the predictor configuration to be varied in isolation.

### 2.3 Future range and control-aligned collision prediction

FutureRay [1] is the immediate basis of this work. It predicts, for each of a set of directions derived from the controller's own candidate commands, how much free range the robot will have along that predicted motion within a horizon  $H$ . Because the queries follow the arcs the controller would actually execute, the planner can compare candidate commands by a quantity that already accounts for curvature and speed, rather than by clearance along straight sensor rays. FutureRay treats the horizon, the number of points evaluated along each predicted path, and the number of control-aligned directions as design constants. Our study keeps FutureRay's formulation of control-aligned future range and its integration into a sampling-based planner, and asks a narrower question: how these three constants should be chosen, and whether the horizon should change with speed and terrain. We do not propose a new predictor architecture.

Learned collision prediction along candidate actions predates FutureRay in wheeled and aerial navigation. Kahn et al. [28] introduced generalized computation graphs that predict collision probabilities along action sequences over a fixed horizon, and BADGR [27] learned to predict collisions, bumpiness, and position from self-supervised real-world data for a fixed set of action sequences. Both make the prediction horizon a hyperparameter of the model, and neither studies its effect at the speeds of agile quadrupeds. Bansal et al. [33] combined a learned waypoint predictor with model-based trajectory generation, and Loquercio et al. [32] trained a policy that outputs collision-free trajectories for high-speed flight, where the look-ahead of the predicted trajectories similarly trades reaction time against prediction quality.

### 2.4 Horizon selection in local planning

The horizon question is familiar from model predictive and sampling-based planning. Model predictive path integral control [29] samples control sequences over a fixed horizon and weights them by cost; its horizon must cover the dynamics relevant to the task while keeping sample variance manageable.

Planners for fast aerial vehicles confront the same tension between the speed-dependent stopping distance and the reliability of the map beyond the sensed region: FASTER [30] plans a fast trajectory into unknown space while maintaining a safe back-up trajectory inside known free space, and EGO-Planner [31] reduces computation so that replanning can be frequent enough to compensate for short look-ahead. These works treat look-ahead length and replanning rate as coupled design choices; our rule makes the analogous choice for a learned range predictor, using data rather than an analytic model to locate the trade-off. The FASTER treatment of unknown space is also related to our occluded-turn failure: a future-range query that extends behind an occluder reports range in space that is not known to be free.

## 2.5 Traversability and terrain estimation

Terrain estimation informs both locomotion and planning. Frey et al. [26] learned visual traversability online from proprioceptive feedback, and the adaptation module of RMA [5] infers terrain-related extrinsics from proprioceptive history. Art-Planner [22] used learned reachability and foothold scores in a sampling-based planner for field deployment of ANYmal. Field systems built for the DARPA Subterranean Challenge, including the CERBERUS [39] and CoSTAR [40, 38] teams, operated legged robots in degraded, occluded environments and adjusted planning behavior to terrain and risk. Our terrain estimator is deliberately simpler than these systems: a three-class proprioceptive classifier is sufficient for indexing a horizon table, and its low cost is part of what keeps the rule lightweight.

## 2.6 Simulation and sim-to-real

Our ablation runs in Isaac Lab, which builds on the GPU-parallel physics of Isaac Gym [36] and the Orbit framework [35]; massively parallel simulation [2] is what makes a sweep over configurations, speeds, terrains, scenes, and seeds tractable. Domain randomization [37] and the teacher-student and adaptation methods above [4, 5] are the standard tools for transferring such policies to hardware. Cross-robot navigation models such as GNM [34] show that navigation behavior can be shared across platforms when interfaces are standardized; the control-aligned range interface of FutureRay is one such interface, and our results characterize how its main parameter should be set for one platform.

## 03 Problem Formulation

### 3.1 Robot, commands, and environment

The robot state at planning step  $t$  is  $x_t = (p_t, \theta_t, \nu_t)$ , with planar base position  $p_t \in \mathbb{R}^2$ , yaw  $\theta_t$ , and body-frame twist  $\nu_t$ . The planner issues a body-frame velocity command  $u = (v_x, v_y, \omega)$  to a fixed locomotion policy  $\pi_{\text{loco}}$  that tracks it. The environment is described by an occupied set  $\mathcal{O} \subset \mathbb{R}^2$  (obstacles projected to the plane at body height) and a terrain class  $\tau \in \mathcal{T} = \{\text{flat}, \text{mixed}, \text{rough}\}$ . The robot receives an observation  $o_t$  comprising an exteroceptive point cloud and proprioceptive history, and must reach a goal  $g$  without collision.

### 3.2 Control-aligned directions

Following FutureRay [1], the planner evaluates a finite set of candidate commands  $\mathcal{U}_K = \{u^{(k)}\}_{k=1}^K$ , where  $K$  is the direc-

tion count. For a command held constant over the horizon, the predicted base path is

$$p^{(k)}(s) = p_t + \int_0^s R(\theta_t + \omega^{(k)}\sigma) \begin{bmatrix} v_x^{(k)} \\ v_y^{(k)} \end{bmatrix} d\sigma, \quad (1)$$

for  $s \in [0, H]$ , where  $R(\cdot)$  is the planar rotation matrix and  $s$  is time along the prediction. The path length is  $L_k(H) = \|v^{(k)}\|H$ , with  $\|v^{(k)}\|$  the planar speed of command  $k$ .

### 3.3 Future range

Let  $\mathcal{B}(p, \theta)$  denote the robot's collision footprint at pose  $(p, \theta)$  and let  $\ell$  denote arc length along  $p^{(k)}$ . The ground-truth future range of direction  $k$  at horizon  $H$  is the free arc length before the swept footprint first meets an obstacle, truncated at the horizon:

$$r_k(H) = \min\{L_k(H), \ell_k^c\}, \quad (2)$$

where the first-contact arc length is

$$\ell_k^c = \inf \{\ell \in [0, L_k(H)] : \mathcal{B}(p^{(k)}(\ell), \theta^{(k)}(\ell)) \cap \mathcal{O} \neq \emptyset\}, \quad (3)$$

with  $\ell_k^c = \infty$  if no contact occurs. A predictor  $f_\phi$  estimates  $\hat{r}_k = f_\phi(o_t, x_t, u^{(k)}; H)$  for all  $k$  in one forward pass. The predictor evaluates the path at  $M$  query points  $s_{k,m} = mH/M$ ,  $m = 1, \dots, M$ , with

$$M = \rho M_0, \quad (4)$$

where  $M_0$  is the nominal number of queries per direction and  $\rho \geq 1$  is the query-density multiplier. A predictor configuration is the triple  $c = (H, \rho, K)$ .

### 3.4 Ablation objective

For a fixed navigation stack, let  $\text{SR}(c; v, \tau)$ ,  $E_{95}(c; v, \tau)$ ,  $B(c; v, \tau)$ , and  $\bar{v}(c; v, \tau)$  denote, respectively, the success rate, the 95th-percentile tracking error, the number of braking events per kilometer, and the mean speed maintained, when configuration  $c$  is used at reference speed  $v$  on terrain  $\tau$ . The ablation estimates these quantities over a grid  $\mathcal{C} \times \mathcal{V} \times \mathcal{T}$  with  $\mathcal{V} = \{1, 2, 3, 4, 5\}$  m/s and horizon set  $\mathcal{H} = \{0.25, 0.5, 1.0, 2.0\}$  s. The selection problem is then to find a map  $h : \mathcal{V} \times \mathcal{T} \rightarrow \mathcal{H}$  that, when used online with an estimated terrain class  $\hat{\tau}$ , performs at least as well as any fixed horizon.

## 04 Method

### 4.1 Fixed navigation stack

All configurations share the same navigation stack, which has three layers.

**Locomotion.** A velocity-tracking reinforcement-learning policy  $\pi_{\text{loco}}$  for the Go2, trained once in Isaac Lab with terrain curricula and domain randomization in the style of [2, 4], maps proprioceptive history and the command  $u$  to joint position targets. It runs at 50 Hz and is not retrained for any configuration.

**Local planner.** At each planning step the planner scores the  $K$  control-aligned candidates using the predicted future range. With  $d_{\text{stop}}(v)$  the stopping distance at speed  $v$ ,

$$d_{\text{stop}}(v) = v t_\ell + \frac{v^2}{2a_{\text{max}}} + d_0, \quad (5)$$

where  $t_\ell$  is the end-to-end latency from sensing to actuation,  $a_{\max}$  the deceleration the locomotion policy can reliably track, and  $d_0$  a fixed clearance margin, the cost of candidate  $k$  is

$$J_k = -w_g \frac{(g - p_t)^\top \Delta p^{(k)}}{\|g - p_t\|} + w_\omega |\omega^{(k)}| + w_r \exp\left(-\frac{\hat{r}_k - d_{\text{stop}}(\|v^{(k)}\|)}{\ell_r}\right), \quad (6)$$

where  $\Delta p^{(k)} = p^{(k)}(H) - p_t$  is the predicted displacement,  $w_g, w_\omega, w_r$  are fixed weights, and  $\ell_r$  sets how sharply the cost rises as the predicted range approaches the stopping distance. The planner selects  $k^* = \arg \min_k J_k$ .

*Speed governor and safety.* The commanded speed is limited so that the robot can stop within the predicted range of the selected direction. Inverting (5),

$$v_{\text{cmd}} = \min\left(v_{\text{ref}}, a_{\max}\left(-t_\ell + \sqrt{t_\ell^2 + 2(\hat{r}_{k^*} - d_0)/a_{\max}}\right)\right), \quad (7)$$

with  $v_{\text{cmd}} = 0$  when  $\hat{r}_{k^*} \leq d_0$ . A braking event is counted whenever the governor reduces the commanded speed by more than a fixed threshold within one planning cycle. Weights, thresholds,  $t_\ell$ ,  $a_{\max}$ , and  $d_0$  are identical across all configurations.

#### 4.2 Predictor parameterization

The future-range predictor follows FutureRay [1]: a point-cloud encoder produces a local feature map, each query point  $p^{(k)}(s_{k,m})$  is embedded with its normalized time  $s_{k,m}/H$  and the command  $u^{(k)}$ , and a per-direction head aggregates the  $M$  query features into a range estimate normalized by  $L_k(H)$ . For each configuration  $c$  a separate predictor is trained on the same simulated data with the same schedule, minimizing

$$\mathcal{L}(\phi) = \sum_{k=1}^K \left[ \ell_\delta \left( \frac{\hat{r}_k - r_k}{L_k(H)} \right) + \beta \frac{[\hat{r}_k - r_k]_+}{L_k(H)} \right], \quad (8)$$

where  $\ell_\delta$  is the Huber loss and the second term, with  $[z]_+ = \max(z, 0)$  and  $\beta > 0$ , penalizes overestimation of free range more than underestimation, since overestimation is the safety-relevant error. Normalizing by  $L_k(H)$  keeps the loss scale comparable across horizons.

The *present-only* baseline replaces  $\hat{r}_k$  by the clearance measured along the straight ray from the current pose in the initial direction of  $v^{(k)}$ , truncated at the same maximum length used by the planner; it uses no rollout and corresponds to discarding the control-aligned future component.

#### 4.3 Why the horizon matters

The horizon enters the stack through the truncation in (2). Because  $\hat{r}_k \leq L_k(H) = vH$  at speed  $v$ , the governor (7) cannot command the reference speed unless  $vH \geq d_{\text{stop}}(v)$ , which yields the truncation bound

$$H \geq H_{\text{trunc}}(v) = t_\ell + \frac{v}{2a_{\max}} + \frac{d_0}{v}. \quad (9)$$

Because the planner replans every cycle, violating (9) need not cause failure, but it makes the range signal saturate in open space, which shows up as unnecessary braking and lost speed. On rough terrain a second lower bound appears: the locomotion policy needs advance notice to change gait or posture, so the

horizon must also exceed a terrain-dependent transition time  $T_\tau$ , giving  $H \geq \max(H_{\text{trunc}}(v), T_\tau)$ . This bound is largest at low speed on rough terrain through  $T_\tau$ , and at high speed through the  $v/(2a_{\max})$  term.

Long horizons carry an opposite cost. The predicted path (1) assumes a constant command, while the executed path deviates from it as the planner replans and the locomotion policy tracks imperfectly; the deviation grows with  $H$ . In addition, the farther end of a long path increasingly lies outside the sensed region, especially behind occluders, where the predictor must extrapolate. Writing the prediction error as  $\epsilon_k(H) = \hat{r}_k - r_k$ , both effects increase  $\mathbb{E}|\epsilon_k(H)|$  with  $H$ , and an overestimate behind an occluder is particularly harmful because the planner is drawn toward directions whose apparent range lies in unobserved space. The useful horizon is therefore bracketed from below by truncation and gait-transition requirements and from above by extrapolation error, and both brackets move with speed and terrain. The ablation locates this bracket empirically rather than relying on the constants in (9).

#### 4.4 Ablation design

The study varies  $c = (H, \rho, K)$  over  $H \in \mathcal{H}$ , query-density multipliers  $\rho$  starting from the nominal value  $\rho = 1$ , and several direction counts  $K$ , while everything in the navigation stack above is held fixed. Each configuration is evaluated at every reference speed in  $\mathcal{V}$  and on every terrain class in  $\mathcal{T}$ . The scene set, seeds, and trial counts are described in Section 5.1.

#### 4.5 Horizon-selection rule

The rule is fitted from the simulated grid. For each speed-terrain cell, a scalar score combines the three outcome measures,

$$S(H; v, \tau) = \text{SR}(H; v, \tau) - \lambda_B B(H; v, \tau) - \lambda_E E_{95}(H; v, \tau), \quad (10)$$

with the query density and direction count fixed to the values that maximize the score aggregated over all cells, and  $\lambda_B, \lambda_E > 0$  trading safety-relevant braking and tracking tail against success. The table entry is

$$h(v, \tau) = \min \left\{ H \in \mathcal{H} : S(H; v, \tau) \geq \max_{H' \in \mathcal{H}} S(H'; v, \tau) - \varepsilon \right\}, \quad (11)$$

that is, the shortest horizon whose score is within a tolerance  $\varepsilon$  of the best. The tie-break toward short horizons is a deliberate design choice: when two horizons are statistically indistinguishable in simulation, the shorter one depends less on extrapolation into unobserved space, which the analysis above identifies as the main failure mechanism of long horizons.

Online, the rule looks up  $h(\tilde{v}, \hat{\tau}_t)$ , where  $\tilde{v}$  is the reference speed rounded to the nearest grid speed and  $\hat{\tau}_t$  is the estimated terrain class. To avoid chattering between horizons when the speed or terrain estimate is near a cell boundary, a new horizon is adopted only after the lookup has returned it for  $N_h$  consecutive planning cycles. The trained predictors for all horizons in  $\mathcal{H}$  are kept loaded, so a horizon switch only changes which predictor is evaluated in the next planning cycle and requires no re-initialization.

#### 4.6 Terrain estimation

The terrain class is estimated from proprioception. Over a sliding window, a feature vector  $\varphi_t$  is formed from the standard

deviations of base roll and pitch rates, the mean foot-slip speed during stance, and the standard deviation of joint torques. A linear softmax classifier gives  $q_t(\tau) = \text{softmax}(W\varphi_t + b)_\tau$ , which is smoothed and thresholded as

$$\pi_t = (1 - \alpha) \pi_{t-1} + \alpha q_t, \quad \hat{\tau}_t = \arg \max_{\tau \in \mathcal{T}} \pi_t(\tau), \quad (12)$$

with smoothing factor  $\alpha \in (0, 1)$ . The classifier is trained on labeled simulated rollouts with the same locomotion policy. The rule therefore consists of a three-class linear classifier, a first-order filter, a table lookup, and a hysteresis counter; Section 5.7 reports its measured decision latency.

## 05 Experiments

### 5.1 Setup and platform

All simulated experiments use a Unitree Go2 model in Isaac Lab. The scene set contains 12 scene families drawn from five categories—open, cluttered, narrow, occluded, and high-speed corridor—each instantiated with 5 seeds and evaluated with 25 trials per scene–seed instance, i.e.,  $12 \times 5 = 60$  scene–seed instances and  $60 \times 25 = 1,500$  trials per pass over the scene set. Reference speeds range from 1 to 5 m/s in 1 m/s steps, and terrain is drawn from three classes (flat, mixed, rough). The full navigation stack is fixed; only the future-range predictor’s horizon  $H$ , query density, and direction count vary. The selected rule is additionally run on a physical Go2 with 8 runs per speed.

#### REPRODUCIBILITY NOTE

The locomotion policy, planner weights, governor constants, and safety thresholds are shared by every configuration in every table. Differences between rows therefore arise from the predictor configuration and, for the selected rule, from online horizon switching.

### 5.2 Metrics

*Success rate* is the percentage of trials in which the robot reaches the goal without collision or timeout. *Tracking error p95* is the 95th percentile of the deviation between the commanded reference and the executed base trajectory, in centimeters; it captures tail tracking quality rather than average behavior. *Braking events per km* counts governor-triggered speed reductions, as defined in Section 4, normalized by distance traveled. *Mean speed maintained* is the average base speed over the traversed distance in m/s.

### 5.3 Main comparison

Table 1 compares the selected rule with three fixed horizons and the present-only baseline. The rule achieves a success rate of 92.3%, which is 11.2 points above the best fixed horizon in this table ( $H = 2.0$  s, 81.1%) and 11.4 points above the present-only baseline (80.9%). It also has the lowest tracking error p95 (13.8 cm, against 14.7 cm for the best baselines), the fewest braking events (2.1 per km, against 2.2–2.3), and the highest mean speed maintained (3.30 m/s, 0.25 m/s above the fastest fixed horizon,  $H = 1.0$  s at 3.05 m/s).

Among the fixed horizons in Table 1, success increases with horizon length, from 77.6% at  $H = 0.25$  s to 81.1% at  $H = 2.0$  s, whereas mean speed peaks at  $H = 1.0$  s. The present-only baseline (80.9%) is above  $H = 0.25$  s and  $H = 1.0$  s on success and within 0.2 points of  $H = 2.0$  s. In this table,

then, a fixed future-range horizon does not improve clearly on present-time range, and the benefit of future range appears only when the horizon is adapted.

### 5.4 Ablation

Table 2 reports the success rate of the selected rule and of single-component variants. Replacing the rule with a fixed horizon lowers success by 1.3 points at  $H = 0.25$  s (91.0%), 5.8 points at  $H = 1.0$  s (86.5%), and 7.5 points at  $H = 2.0$  s (84.8%). Reverting the query density to the nominal multiplier ( $\rho = 1$ , “query density 1x”) lowers success by 9.7 points to 82.6%, the largest drop in the table. Within this table, success decreases monotonically with fixed horizon length, and the query density matters more than the choice among fixed horizons.

### 5.5 Horizon sweep

Table 3 reports the horizon sweep. Success peaks at  $H = 0.5$  s (92.3%) and declines on both sides: by 3.9 points at  $H = 0.25$  s (88.4%), 0.7 points at  $H = 1.0$  s (91.6%), and 2.6 points at  $H = 2.0$  s (89.7%). The same horizon minimizes both cost measures: tracking error p95 is 13.8 cm at  $H = 0.5$  s against 17.5 cm at  $H = 2.0$  s, and braking events fall from 2.9 per km at  $H = 0.25$  s to 2.1 at  $H = 0.5$  s before rising to 2.6 at  $H = 1.0$  s and 3.4 at  $H = 2.0$  s. The decline is asymmetric in the two cost measures: moving from 0.5 s to 2.0 s raises tracking error p95 by 3.7 cm and braking by 1.3 events per km, whereas moving from 0.5 s to 0.25 s raises them by 2.4 cm and 0.8 events per km. Success shows the opposite asymmetry, dropping more at 0.25 s (3.9 points) than at 2.0 s (2.6 points). Together these are consistent with the analysis of Section 4: short horizons pay a truncation cost that shows mainly in success, and long horizons pay an extrapolation cost that shows mainly in tracking tail and braking.

The sweep is the one-dimensional slice of the accuracy–cost surface along  $H$  that the record tabulates. Its  $H = 0.5$  s row is identical in all three metrics to the selected rule’s row in Table 1. Within the sweep, therefore, the rule matches, but does not exceed, the best single fixed horizon.

### 5.6 Consistency across tables

The three tables report success rates for fixed horizons that do not agree with each other. For  $H = 0.25$  s the recorded values are 77.6% (Table 1), 91.0% (Table 2), and 88.4% (Table 3); for  $H = 1.0$  s they are 79.9%, 86.5%, and 91.6%; for  $H = 2.0$  s they are 81.1%, 84.8%, and 89.7%. The orderings also differ: success rises with horizon in Table 1, falls with horizon in Table 2, and peaks at 0.5 s in Table 3. The experimental record does not identify which scenes, speeds, terrains, query densities, or direction counts underlie each table, so these differences cannot be attributed to a specific cause, and the tables should not be compared row-for-row. The conclusions drawn in this paper are restricted to comparisons within a table. Across all three tables, the selected rule’s success rate (92.3%) is at least as high as every fixed-horizon entry; it is strictly higher in Tables 1 and 2 and equal to the  $H = 0.5$  s entry in Table 3.

### 5.7 Failure cases and hardware

*Occluded turns at 4 m/s.* On occluded turns at 4 m/s,  $H = 2.0$  s is the worst horizon, with 4.7 braking events per km. This is 1.3 events per km more than the 3.4 per km of  $H = 2.0$  s in the sweep aggregate, and more than double the 2.1 per km of  $H = 0.5$  s there. The mechanism matches the extrapolation argument of Section 4: at 4 m/s a 2.0 s horizon extends each can-

**Table 1:** Main comparison in simulation (Unitree Go2, Isaac Lab). Higher is better for success rate and mean speed; lower is better for tracking error and braking.

| Method                             | Success rate (%) | Tracking error p95 (cm) | Braking events per km | Mean speed maintained (m/s) |
|------------------------------------|------------------|-------------------------|-----------------------|-----------------------------|
| $H = 0.25$ s fixed                 | 77.6             | 15.8                    | 2.2                   | 2.84                        |
| $H = 1.0$ s fixed                  | 79.9             | 14.8                    | 2.3                   | 3.05                        |
| $H = 2.0$ s fixed                  | 81.1             | 14.7                    | 2.2                   | 2.92                        |
| No future range (present only)     | 80.9             | 14.7                    | 2.3                   | 2.93                        |
| <b>Ours (selected online rule)</b> | <b>92.3</b>      | <b>13.8</b>             | <b>2.1</b>            | <b>3.30</b>                 |

**Table 2:** Ablation of the selected rule (simulation).

| Variant              | Success rate (%) |
|----------------------|------------------|
| Selected online rule | 92.3             |
| $H = 0.25$ s fixed   | 91.0             |
| $H = 1.0$ s fixed    | 86.5             |
| $H = 2.0$ s fixed    | 84.8             |
| Query density 1x     | 82.6             |

**Table 3:** Horizon sweep (simulation).

| $H$ (s) | Success (%) | Track. err. p95 (cm) | Braking / km |
|---------|-------------|----------------------|--------------|
| 0.25    | 88.4        | 16.2                 | 2.9          |
| 0.5     | 92.3        | 13.8                 | 2.1          |
| 1.0     | 91.6        | 14.9                 | 2.6          |
| 2.0     | 89.7        | 17.5                 | 3.4          |

didate path 8 m ahead, and on a turn much of that path lies behind the occluding geometry. The predictor then reports range in space it cannot observe, so the planner is drawn toward directions whose apparent range lies behind the occluder (the range “chases” the unobserved region), and the governor brakes when the occluded space resolves. The rule does not observe occlusion directly; it reduces this failure only to the extent that the speed–terrain table assigns shorter horizons at high speed.

*Rough terrain at 1 m/s.* On rough terrain at 1 m/s,  $H = 0.25$  s loses 4.1 success-rate points. At 1 m/s a 0.25 s horizon covers only 0.25 m, which is short relative to the Go2’s body length; the controller does not receive range information early enough to plan gait changes, which corresponds to the gait-transition bound  $T_\tau$  of Section 4 rather than to the stopping-distance bound. This case shows why a speed-only rule is insufficient: the lower bound on  $H$  is driven by speed on flat ground but by terrain at low speed on rough ground.

*Latency and hardware.* The rule’s online adaptation adds 0.4 ms of decision latency. On the physical Go2, with 8 runs per speed, the rule selected horizons between 0.5 and 1.0 s across 8/8 runs, as expected from the simulated table. The record documents horizon selection on hardware but not hardware success rates, tracking errors, or braking counts, so the hardware experiment validates that the rule behaves as designed online, not that the simulated gains transfer.

## 06 Limitations

Several limitations bound the conclusions. First, as described above, the fixed-horizon entries disagree across the three recorded tables and the record does not specify the evaluation subset behind each table. The 11.2-point gain of the rule over the best fixed horizon in Table 1 is therefore not corroborated by the sweep, in which the rule only matches  $H = 0.5$  s. A practitioner who has already tuned a single horizon near 0.5 s should not expect the gain in Table 1.

Second, although the ablation varies query density and direction count, the recorded results tabulate only one density variant (“query density 1x”) and no direction-count results. The accuracy–cost surface is therefore reported only along the horizon axis, and the compute cost of the predictor configurations is not quantified beyond the 0.4 ms latency of the rule itself.

Third, the hardware validation reports which horizons the rule selected but not navigation outcomes, so sim-to-real transfer of the performance gains is untested. Fourth, the rule conditions on speed and terrain only. The occluded-turn failure suggests that an occlusion-aware input, for example the fraction of each candidate path that lies outside the sensed region, would address the dominant failure of long horizons more directly than a speed–terrain table. Fifth, all results are for one platform, one locomotion policy, and one planner; the numerical location of the best horizon depends on the latency, deceleration limit, and clearance margin in (9), and will differ for other robots and stacks. Finally, the record does not include confidence intervals or per-cell variability, so small differences, such as the 0.7-point gap between  $H = 0.5$  s and  $H = 1.0$  s in the sweep, should not be overinterpreted.

## 07 Conclusion

This paper studied how much future range a control-aligned range predictor should provide to an agile quadruped navigation stack. With the stack held fixed on a simulated Unitree Go2, the recorded horizon sweep places the best horizon at 0.5 s, where success, tail tracking error, and braking are simultaneously best, and shows that long horizons are most harmful on occluded turns at high speed, while very short horizons fail on rough terrain at low speed. These two failure regimes correspond to the extrapolation and gait-transition bounds derived in the method and motivate a horizon that depends on both speed and terrain. A lightweight speed–terrain lookup with a proprioceptive terrain estimator adds 0.4 ms of latency, outperforms every fixed horizon and the present-only baseline in the main comparison, and selected 0.5–1.0 s horizons on a physical Go2. The evidence that it outperforms a well-chosen fixed horizon is weaker: in the sweep it matches  $H = 0.5$  s exactly, and the fixed-horizon results are inconsistent across tables. Resolving that inconsistency with a single documented evaluation, adding occlusion awareness to the rule, and measuring navigation outcomes on hardware are the most direct next steps.

## References

- [1] T. Zang, S. Wang, H. Huang, and S. Cheng. FutureRay: Control-Aligned Future Range for Agile Quadruped Navigation. arXiv:2608.01197, 2026.
- [2] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning. arXiv:2109.11978, 2021.

- [3] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning Robust Perceptive Locomotion for Quadrupedal Robots in the Wild. arXiv:2201.08117, 2022.
- [4] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning Quadrupedal Locomotion over Challenging Terrain. arXiv:2010.11251, 2020.
- [5] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid Motor Adaptation for Legged Robots. arXiv:2107.04034, 2021.
- [6] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning Agile and Dynamic Motor Skills for Legged Robots. arXiv:1901.08652, 2019.
- [7] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-Real: Learning Agile Locomotion For Quadruped Robots. arXiv:1804.10332, 2018.
- [8] 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. arXiv:2202.05481, 2022.
- [9] Y. Kim, H. Oh, J. Lee, J. Choi, G. Ji, M. Jung, D. Youm, and J. Hwangbo. Not Only Rewards But Also Constraints: Applications on Legged Robot Locomotion. arXiv:2308.12517, 2023.
- [10] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid Locomotion via Reinforcement Learning. arXiv:2205.02824, 2022.
- [11] G. B. Margolis and P. Agrawal. Walk These Ways: Tuning Robot Control for Generalization with Multiplicity of Behavior. arXiv:2212.03238, 2022.
- [12] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged Locomotion in Challenging Terrains using Egocentric Vision. arXiv:2211.07638, 2022.
- [13] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme Parkour with Legged Robots. arXiv:2309.14341, 2023.
- [14] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot Parkour Learning. arXiv:2309.05665, 2023.
- [15] D. Hoeller, N. Rudin, D. Sako, and M. Hutter. ANYmal Parkour: Learning Agile Navigation for Quadrupedal Robots. arXiv:2306.14874, 2023.
- [16] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter. Advanced Skills by Learning Locomotion and Local Navigation End-to-End. arXiv:2209.12827, 2022.
- [17] D. Hoeller, L. Wellhausen, F. Farshidian, and M. Hutter. Learning a State Representation and Navigation in Cluttered and Dynamic Environments. arXiv:2103.04351, 2021.
- [18] Z. Fu, A. Kumar, A. Agarwal, H. Qi, J. Malik, and D. Pathak. Coupling Vision and Proprioception for Navigation of Legged Robots. arXiv:2112.02094, 2021.
- [19] 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. arXiv:2011.12255, 2020.
- [20] S. Kareer, N. Yokoyama, D. Batra, S. Ha, and J. Truong. ViNL: Visual Navigation and Locomotion Over Obstacles. arXiv:2210.14791, 2022.
- [21] Y. Kim, C. Kim, and J. Hwangbo. Learning Forward Dynamics Model and Informed Trajectory Sampler for Safe Quadruped Navigation. arXiv:2204.08647, 2022.
- [22] L. Wellhausen and M. Hutter. ArtPlanner: Robust Legged Robot Navigation in the Field. arXiv:2303.01420, 2023.
- [23] J. Lee, M. Bjelonic, A. Reske, L. Wellhausen, T. Miki, and M. Hutter. Learning Robust Autonomous Navigation and Locomotion for Wheeled-Legged Robots. arXiv:2405.01792, 2024.
- [24] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter. Perceptive Locomotion through Nonlinear Model Predictive Control. arXiv:2208.08373, 2022.
- [25] D. Hoeller, N. Rudin, C. Choy, A. Anandkumar, and M. Hutter. Neural Scene Representation for Locomotion on Structured Terrain. arXiv:2206.08077, 2022.
- [26] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter. Fast Traversability Estimation for Wild Visual Navigation. arXiv:2305.08510, 2023.
- [27] G. Kahn, P. Abbeel, and S. Levine. BADGR: An Autonomous Self-Supervised Learning-Based Navigation System. arXiv:2002.05700, 2020.
- [28] G. Kahn, A. Villafior, B. Ding, P. Abbeel, and S. Levine. Self-supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation. arXiv:1709.10489, 2017.
- [29] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou. Information Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving. arXiv:1707.02342, 2017.
- [30] J. Tordesillas, B. T. Lopez, and J. P. How. FASTER: Fast and Safe Trajectory Planner for Flights in Unknown Environments. arXiv:1903.03558, 2019.
- [31] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao. EGO-Planner: An ESDF-free Gradient-based Local Planner for Quadrotors. arXiv:2008.08835, 2020.
- [32] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. Learning High-Speed Flight in the Wild. arXiv:2110.05113, 2021.
- [33] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin. Combining Optimal Control and Learning for Visual Navigation in Novel Environments. arXiv:1903.02531, 2019.
- [34] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. GNM: A General Navigation Model to Drive Any Robot. arXiv:2210.03370, 2022.
- [35] M. Mittal et al. Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments. arXiv:2301.04195, 2023.
- [36] V. Makoviychuk et al. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. arXiv:2108.10470, 2021.
- [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. arXiv:1703.06907, 2017.
- [38] A. Bouman et al. Autonomous Spot: Long-Range Autonomous Exploration of Extreme Environments with Legged Locomotion. arXiv:2010.09259, 2020.
- [39] M. Tranzatto et al. CERBERUS in the DARPA Subterranean Challenge. arXiv:2201.07067, 2022.
- [40] A. Agha et al. NeBula: Quest for Robotic Autonomy in Challenging Environments; TEAM CoSTAR at the DARPA Subterranean Challenge. arXiv:2103.11470, 2021.
- [41] R. Yang, M. Zhang, N. Hansen, H. Xu, and X. Wang. Learning Vision-Guided Quadrupedal Locomotion End-to-End with Cross-Modal Transformers. arXiv:2107.03996, 2021.