

High-Speed Obstacle Negotiation for Agile Quadrupeds with Range-Predictive Control

Carlos Ramirez • Felix Wagner • Ethan Brooks • Ananya Iyer

Legged Robotics Lab, Stanford University, Stanford, CA, USA

PUBLISHED

September 8, 2026

REGISTERED

September 13, 2026

RECORD TYPE

Preprint

ABSTRACT

At running speeds of 4 to 6 m/s, the reaction budget of a quadruped robot is dominated by the lag of its velocity controller rather than by perception, so a controller that reacts to obstacles already present in its range measurements acts too late. This paper presents a range-predictive negotiation controller built on FutureRay, a perception module that predicts future free range along control-aligned directions. The controller uses the predicted range field to pre-shape the forward and yaw velocity commands before a geometric conflict materializes, schedules the pre-shaping lead time from an identified lag model, and applies an explicit hysteretic rule that hands control to a braking policy when the predicted range collapses below the braking envelope. The design space was explored in Isaac Lab with five seeds, and the controller was evaluated on a Unitree Go2 in a controlled course with six obstacle classes, approach speeds from 4 to 6 m/s in 0.5 m/s steps, and 20 trials per class and speed. Compared with a present-range reactive baseline, the controller raises the mean maximum safe speed over obstacle classes from 4.72 to 4.90 m/s, lowers the miss rate from 4.2 to 3.8 percent, and raises braking handoff success from 90.2 to 97.2 percent. Ablations indicate that the handoff rule, the predicted range, and lead-time scheduling each contribute. Failure analysis shows that at 6.0 m/s the occluded gap class is limited by the unobserved range behind the occluder rather than by the controller.

INDEX TERMS quadruped locomotion, high-speed navigation, predictive range perception, velocity pre-shaping, braking handoff, Unitree Go2

KAGG NOTE At running speed the binding constraint is velocity-controller lag, so the command must begin to change before the conflict appears in the present range; prediction helps only when it is wired into the command and into an explicit decision to stop.

01 Introduction

Learned locomotion controllers have made quadruped robots fast. Policies trained in simulation and transferred to hardware now track forward velocity commands of several meters per second [2, 8, 4], and perceptive controllers traverse steps, gaps, and parkour-style obstacles by fusing exteroception with proprioception [6, 11, 14, 15, 16]. Collision-free navigation at the upper end of this speed range is less mature. When a quadruped approaches an obstacle at 4–6 m/s, the question is no longer only whether the obstacle is perceived, but whether the controller can change the body’s velocity in time to negotiate it.

The difficulty is a matter of timing. Between the moment a conflict enters the decision logic and the moment the base actually decelerates or turns, the robot travels a distance that grows linearly with speed: the command must pass through the velocity-tracking policy, which responds with a transport delay and a finite bandwidth. Braking distance grows quadratically, but it is fixed by the gait and the available friction, whereas the lag distance is the part of the budget that a navigation layer can influence. At 4–6 m/s the lag distance dominates this controllable part. A reactive controller that responds to obstacles already present in its range measurements spends this budget after the conflict is detected. A longer sensing range does not by itself solve the problem: the controller must *act* on far-field information early, and for occluded or moving obstacles the present range does not encode the conflict until late.

FutureRay [1] provides a suitable perceptual input for acting early. It predicts the future free range along the robot’s own predicted control-aligned directions, so each query corresponds to a motion the controller can actually execute. This paper asks how such a prediction should be turned into action at running speed. Two answers are developed. First, the predicted range field is used to *pre-shape* the velocity command, that is, to begin decelerating or turning before the geometric conflict materializes, with a lead time scheduled from an identified model of the velocity-controller lag. Second, when the predicted range collapses so that no admissible pre-shaped command remains, an explicit rule hands control to a dedicated braking policy instead of letting the pre-shaping layer attempt a late correction.

The contributions are as follows.

1. A lag-aware formulation of high-speed obstacle negotiation for quadrupeds that decomposes the reaction budget into transport delay, tracking lag, braking distance, and margin, and that defines the maximum safe speed and the miss rate per obstacle class (Section 3).
2. A range-predictive pre-shaping controller that converts the FutureRay predicted range field into conservative linear constraints of a small quadratic program over forward and yaw velocity commands, with a lead-time schedule derived from the lag model (Section 4).
3. An explicit, hysteretic braking handoff rule, triggered by collapse of the predicted range margin, infeasibility of the

pre-shaping program, or rapid range closure, which transfers control to a separately trained braking policy.

4. A hardware evaluation on a Unitree Go2 over six obstacle classes at 4–6 m/s, with 600 trials per method, comparing against a present-range reactive baseline and two further baselines, together with an ablation, a design-speed sweep, and an analysis of recorded failure cases (Section 5).

The measured improvements are moderate in maximum safe speed and miss rate and larger in braking handoff success. Relative to the present-range reactive baseline, the mean maximum safe speed over classes rises from 4.72 to 4.90 m/s, the miss rate falls from 4.2% to 3.8%, and braking handoff success rises from 90.2% to 97.2%. The failure analysis also identifies a regime, the occluded gap at 6.0 m/s, where the limit is set by what can be observed rather than by the controller.

02 Related Work

2.1 Learned agile locomotion and the velocity-tracking layer

Hwangbo et al. [2] trained locomotion policies in simulation with a learned actuator model and transferred them to ANYmal, establishing the sim-to-real recipe on which most subsequent work builds. Lee et al. [3] used a teacher-student scheme to obtain a proprioceptive policy robust to challenging terrain, and Rudin et al. [4] showed that massively parallel simulation reduces training to minutes. RMA [5] adapts to changing dynamics through an online estimate of latent environment parameters, and Ji et al. [7] train a state estimator concurrently with the policy so that body velocity is estimated reliably during dynamic motion. Margolis et al. [8] reached high running speeds on the Mini Cheetah through an adaptive velocity curriculum, Walk These Ways [9] exposes a family of gaits through an extended command interface, and DreamWaQ [10] learns implicit terrain imagination for blind locomotion. A review of the field is given by Ha et al. [17].

These works provide the layer that the present controller commands. The relevant observation for this paper is that a command-tracking policy is, by design, a low-pass element: it is trained to follow velocity commands smoothly and stably, not to respond instantaneously to step changes. The controller developed here does not modify or retrain the locomotion policy. Instead, it identifies the policy’s closed-loop response as a delay plus a first-order lag and plans the command around it.

2.2 Perceptive locomotion and parkour

Miki et al. [6] fuse exteroceptive height samples with proprioception through a belief encoder that learns when to distrust perception, enabling robust hiking on ANYmal. Agarwal et al. [11] and Yang et al. [12] control legged robots directly from egocentric depth, the latter through a 3D neural volumetric memory, and Hoeller et al. [13] learn a neural scene representation for structured terrain. Robot parkour [14, 15] and ANYmal Parkour [16] negotiate high obstacles, gaps, and climbs with agile skills selected or composed from perception. These systems negotiate obstacles at the level of footholds and whole-body motion, generally at forward speeds below the 4–6 m/s range studied here. The present work operates one level higher, at the velocity command issued ahead of an obstacle, and is complementary: a better negotiation skill raises the class-dependent speed that the command must reach at the obstacle, while the pre-shaping controller determines whether that speed is reached in time.

2.3 Learned navigation and collision-free high-speed legged motion

Rudin et al. [18] learn locomotion and local navigation end-to-end, and Hoeller et al. [19] learn a state representation for navigation among dynamic obstacles; Truong et al. [20] learn navigation skills that transfer across legged robots through learned robot embeddings. Kim et al. [22] impose constraints rather than reward shaping to obtain predictable legged behavior. For perception-driven navigation, Frey et al. [23] estimate traversability online from self-supervision, and BADGR [24] learns event probabilities, including collisions, from autonomously collected data.

The closest work in intent is Agile But Safe (ABS) [21], which pairs an agile policy with a recovery policy and switches between them using a learned reach-avoid value computed from the current state and from ray distances predicted from depth. ABS shows that an explicit switch to a recovery behavior is effective at high speed. The present controller differs in two respects. Its switching rule is stated in terms of a predicted range margin against an explicit stopping envelope rather than a learned value, which makes the handoff condition inspectable and tunable. In addition, a pre-shaping layer acts before the switch is needed, so that the braking policy is invoked only when the predicted range leaves no admissible pre-shaped command.

2.4 Control-aligned future range prediction

FutureRay [1] is the perceptual foundation of this work. Conventional range representations, such as ray casts or occupancy grids in the robot frame, describe free space along fixed geometric directions at the current time. FutureRay instead predicts the free range that will be available at future times along directions aligned with the robot’s own predicted controls, so that each query asks how much free range there will be along a motion the robot can actually execute. This alignment matters at high speed, where the reachable set is narrow and a geometric ray toward an open direction may be dynamically unreachable within the horizon.

This paper uses the FutureRay module without modification and does not claim contributions to range prediction. Its contribution is the control layer that consumes the predicted field: a lag model, a lead-time schedule, a constrained pre-shaping program, and a handoff rule. The baseline that uses the predicted range without pre-shaping (Section 5) isolates the value of the predicted field when it is consumed by a reactive rule rather than by a controller designed around it.

2.5 Reactive collision avoidance and safety filters

The dynamic window approach [25] restricts velocity commands to those reachable within one control interval and admissible in the sense that the robot can stop before the nearest obstacle along the resulting arc. The stopping-envelope constraint used here follows the same admissibility logic but accounts for controller lag and evaluates admissibility against predicted rather than present range. Velocity obstacles [26] characterize the set of velocities that lead to collision with moving obstacles; the moving-cart class in this study is handled implicitly, through the predicted range, rather than through an explicit velocity-obstacle construction.

Control barrier functions enforce forward invariance of a safe set through a quadratic program that minimally modifies a nominal command [27, 28]. Hsu et al. [29] unify such

constructions, Hamilton-Jacobi reachability [30, 31], and predictive approaches under the notion of a safety filter that monitors a nominal policy and intervenes with a fallback. The predictive safety filter of Wabersich and Zeilinger [32] certifies a nominal input by verifying that a backup trajectory to a safe terminal set exists. Reachability-based trajectory design [33] and FASTER [34] similarly maintain a fail-safe maneuver that ends in known free space. The handoff rule developed here is a lightweight instance of this pattern: the pre-shaping program plays the role of the filtered nominal controller, the stopping envelope defines the feasibility of the backup, and the braking policy is the backup. Unlike these methods, it makes no formal guarantee, because the predicted range is a learned estimate whose errors are bounded only empirically.

2.6 High-speed planning in aerial and model-based systems

High-speed quadrotor flight confronts the same timing problem. EGO-Planner [35] replans local trajectories at high rates without a full distance field, and Loquercio et al. [36] fly through unstructured environments at high speed with a policy that maps sensor input to trajectories, explicitly accounting for latency in the sensing-to-action loop. Champion-level drone racing [37] and the comparison of optimal control and reinforcement learning for racing [38] show that performance near the actuation limit depends on planning for, rather than reacting to, upcoming geometry. Model-based legged control, from convex MPC on the MIT Cheetah 3 [39] to perceptive nonlinear MPC on ANYmal [40], plans contact forces and body motion over a horizon. The present controller borrows the horizon view of these methods but applies it at the velocity-command level, where the plant is the closed-loop locomotion policy rather than the rigid-body dynamics.

2.7 Simulation infrastructure

The design-space study uses Isaac Lab, the successor of the Orbit framework [42], built on GPU-parallel physics simulation [41]. The braking policy is trained with PPO [43] under domain randomization [44] of friction, payload, and command latency.

03 Problem Formulation

3.1 Commanded motion and controller lag

The robot runs a fixed velocity-tracking locomotion policy, and the navigation layer runs at control period Δt . The navigation layer issues a command $u_t = (v_t^{\text{cmd}}, \omega_t^{\text{cmd}})$ of forward speed and yaw rate. The closed-loop response of the locomotion policy to forward-speed commands is modeled as a transport delay δ followed by a first-order lag with time constant τ_v and an acceleration bound a_{max} :

$$\dot{v}(t) = \text{sat}_{a_{\text{max}}} \left(\frac{v^{\text{cmd}}(t - \delta) - v(t)}{\tau_v} \right), \quad (1)$$

where $v(t)$ is the measured forward speed of the base and $\text{sat}_a(\cdot)$ clips its argument to $[-a, a]$. The yaw channel is modeled in the same form with parameters $(\delta_\omega, \tau_\omega)$. The parameters are identified from step-response logs of the locomotion policy on the course surface. The model is deliberately simple: it is used to schedule the command, not to predict the gait.

When a full stop is required, a braking policy decelerates the robot at a sustained deceleration a_b once it is engaged. The distance consumed from the decision to stop until rest is bounded by the stopping envelope

$$d_{\text{stop}}(v) = v(\delta + \tau_v) + \frac{v^2}{2a_b} + d_0, \quad (2)$$

where the first term is the lag distance, the second is the braking distance, and d_0 is a fixed clearance margin that also absorbs the offset between the range origin and the front of the body. The first term is the one that pre-shaping can remove from the critical path: if the command begins to change $\delta + \tau_v$ earlier, the lag is consumed while the range is still large.

3.2 Predicted range field

At each control step FutureRay [1] provides a predicted range field

$$\hat{R}_t = F_\theta(o_{t-H:t}, u_{t-H:t}) \in \mathbb{R}_{\geq 0}^{(K+1) \times J \times N_b}, \quad (3)$$

where $o_{t-H:t}$ and $u_{t-H:t}$ are the recent observation and command histories and F_θ is the pretrained predictor. The entry $\hat{R}_t(k, j, b)$ is the free range predicted at time $t + k\Delta_h$, measured from the robot's predicted pose at that time along the control-aligned direction $\psi_j(k)$ of candidate command $j \in \{1, \dots, J\}$, within height band b above the ground. The horizon is K steps of duration Δ_h . The three height bands ($N_b = 3$) cover the foot-swing, body, and overhead regions, so that low beams, posts, and high beams constrain different bands. The slice $k = 0$ is the present range, $r_t(j, b) = \hat{R}_t(0, j, b)$, which is what a reactive controller uses. A per-entry uncertainty $\sigma_t(k, j, b)$ accompanies the prediction, from which a conservative field is formed:

$$\underline{R}_t(k, j, b) = \max(0, \hat{R}_t(k, j, b) - \beta \sigma_t(k, j, b)), \quad (4)$$

with confidence multiplier $\beta > 0$. All decisions below use $\underline{R}_t$.

3.3 Negotiation task and metrics

An obstacle of class $c \in \mathcal{C}$ lies on the course ahead of the robot, which approaches it at speed $v \in \mathcal{V}$. Each class has a set of height bands $\mathcal{B}_c$ in which it imposes a range constraint, and a class-appropriate negotiation: passing beside a post, stepping over a low beam with raised foot clearance, passing under a high beam at reduced body height, yielding to or passing behind a moving cart, threading static clutter, and entering an occluded gap. A trial is a *miss* if the robot contacts the obstacle or fails to complete the class-appropriate negotiation. For a class c , let $\hat{P}_{\text{miss}}(c, v)$ be the empirical miss fraction over trials at approach speed v . The maximum safe speed is

$$v_c^* = \sup \{v : \tilde{P}_{\text{miss}}(c, v') \leq \alpha \ \forall v' \leq v\}, \quad (5)$$

where $\tilde{P}_{\text{miss}}$ is the linear interpolation of $\hat{P}_{\text{miss}}$ between adjacent grid speeds and $\alpha = 0.05$, that is, at most one miss in 20 trials. The interpolation produces values between grid points; the reported summary is the mean of v_c^* over the six classes. The *miss rate* of a method is the percentage of misses over all of its trials, pooled across classes and approach speeds. The objective is to maximize v_c^* for every class while keeping the pooled miss rate low, without retraining the locomotion policy or the perception module.

04 Range-Predictive Negotiation Controller

4.1 Overview

The controller is a three-mode supervisor running at Δt on top of the locomotion policy. In *cruise* mode the robot tracks the reference speed v_{ref} and the goal heading. In *pre-shape* mode, entered when a predicted conflict falls within the scheduled lead time, a quadratic program shapes the forward and yaw commands over the prediction horizon. In *brake* mode, entered through the handoff rule, a braking policy takes over until the predicted margin recovers. Every mode consumes the same conservative field $\underline{R}_t$, so the transitions between them are driven by a single quantity, the predicted range margin.

4.2 Direction selection and conflict prediction

For the goal heading ψ_g the controller first selects the control-aligned direction that best combines predicted clearance and goal progress:

$$j_t^* = \arg \max_j \left[\min_{k,b} \underline{R}_t(k, j, b) - \lambda_\psi |\psi_j(K) - \psi_g| \right], \quad (6)$$

with heading weight λ_ψ in meters per radian. A hysteresis term, omitted for brevity, penalizes switching j^* between consecutive steps unless the clearance gain exceeds a threshold. Along the selected direction, the predicted margin at horizon step k for an assumed speed profile $\{v_k\}$ is

$$m_t(k) = \min_{b \in \mathcal{B}} \underline{R}_t(k, j_t^*, b) - d_{\text{stop}}(v_k), \quad (7)$$

which is negative when, at time $t + k\Delta_h$, the robot could no longer stop within the predicted free range. Here $\mathcal{B}$ is the set of all bands; the class-specific subsets $\mathcal{B}_c$ determine which negotiation maneuver and speed cap apply. Evaluating $m_t(k)$ with the unshaped profile $v_k = v_{\text{ref}}$ gives the predicted time to conflict

$$T_c = \Delta_h \cdot \min\{k \in \{0, \dots, K\} : m_t(k) < 0\}, \quad (8)$$

with $T_c = \infty$ if the margin stays nonnegative over the horizon. A reactive controller observes the conflict only once $m_t(0) < 0$; the predicted field exposes it up to $K\Delta_h$ earlier when the conflict is predictable from the history.

4.3 Lead-time scheduling

Pre-shaping must begin early enough that the lag in (1) is consumed before the conflict. Under the model (1), driving the speed from v_t to within a tolerance ε_v of a target v_c takes $\delta + \tau_v \ln(|v_t - v_c|/\varepsilon_v)$ when the acceleration bound is inactive, and at least $\delta + |v_t - v_c|/a_{\text{max}}$ when it is active. The scheduled lead time takes the larger of the two:

$$L(v_t, v_c) = \delta + \max\left(\tau_v \ln \max\left(1, \frac{|v_t - v_c|}{\varepsilon_v}\right), \frac{|v_t - v_c|}{a_{\text{max}}}\right), \quad (9)$$

where $v_c = \bar{v}_c(h, \rho)$ is the negotiation speed cap for the perceived class, obstacle height h , and visible range ρ . The cap is stored as a lookup table obtained from the Isaac Lab design-space study, in which negotiation success was characterized as a function of obstacle height and field-of-view range. Pre-shaping is triggered when

$$T_c \leq L(v_t, v_c) + T_s, \quad (10)$$

where T_s is a settling allowance during which the speed should already be at the cap before the negotiation begins. The schedule is the mechanism that converts predicted range into earlier action; without it, a fixed trigger either fires too late at high approach speed or too early at low speed, where it needlessly reduces speed.

4.4 Pre-shaping program

Once triggered, the controller solves, at every step, a quadratic program over the horizon in the planned commands $\{v_k^{\text{cmd}}, \omega_k^{\text{cmd}}\}$ and the resulting predicted speeds $\{v_k\}$:

$$\begin{aligned} \min_{\{v_k^{\text{cmd}}, \omega_k^{\text{cmd}}\}} \quad & \sum_{k=0}^K \left[(v_k - v_{\text{ref}})^2 + \lambda_a (v_k^{\text{cmd}} - v_{k-1}^{\text{cmd}})^2 \right. \\ & \left. + \lambda_\omega (\omega_k^{\text{cmd}} - \omega_k^*)^2 \right] \\ \text{s.t.} \quad & v_{k+1} = v_k + \frac{\Delta_h}{\tau_v} (v_{k-n_\delta}^{\text{cmd}} - v_k), \\ & |v_{k+1} - v_k| \leq a_{\text{max}} \Delta_h, \\ & c_1 v_k + d_0 \leq \min_{b \in \mathcal{B}} \underline{R}_t(k, j_t^*, b), \\ & v_k \leq \bar{v}_c \quad \forall k \geq k_c, \quad 0 \leq v_k^{\text{cmd}} \leq v_{\text{ref}}. \end{aligned} \quad (11)$$

Here $n_\delta = \lceil \delta / \Delta_h \rceil$ is the delay in horizon steps, with commands before $k = 0$ fixed to those already issued; ω_k^* is the yaw rate that realizes direction j_t^* ; $k_c = T_c / \Delta_h$ is the horizon index of the predicted conflict; and $\lambda_a, \lambda_\omega$ weight command smoothness and heading tracking. The yaw channel enters the objective only, with its own lag model used to compute ω_k^* ; this keeps the program convex and small.

The stopping envelope (2) is quadratic in v_k . Because $0 \leq v_k \leq v_{\text{ref}}$, the chord bound $v_k^2 \leq v_{\text{ref}} v_k$ yields the linear, conservative surrogate used in (11),

$$d_{\text{stop}}(v_k) \leq c_1 v_k + d_0, \quad c_1 = \delta + \tau_v + \frac{v_{\text{ref}}}{2a_b}. \quad (12)$$

Any solution of (11) therefore satisfies the exact envelope with respect to the conservative field. The surrogate is tight near v_{ref} , where it matters, and conservative at low speed, where the cost of conservatism is small. The program has $2(K+1)$ decision variables and is warm-started from the previous solution; only the first command is applied, in receding-horizon fashion.

The choice to pre-shape the command, rather than to cap the reference speed, is deliberate. A global speed cap reduces the lag distance uniformly but sacrifices speed on every segment, including those without a conflict. Pre-shaping reduces speed only where and when the predicted field requires it, and it begins the deceleration while the range is still large, so that the lag term of (2) is spent outside the critical region.

4.5 Braking handoff rule

The pre-shaping program can only redistribute speed along an admissible profile. When the predicted range collapses, for example because an occluder reveals an obstacle or a moving obstacle enters the selected direction, no admissible profile may remain. Letting the program attempt a late correction in this situation produces large, poorly tracked commands. The controller instead monitors the handoff margin

$$\Phi_t = \min_{0 \leq k \leq K_h} \left[\min_{b \in \mathcal{B}} \underline{R}_t(k, j_t^*, b) - d_{\text{stop}}(v_k^{\text{qp}}) \right], \quad (13)$$

Table 1: Controller configuration.

Symbol	Meaning	Setting
Δt	Navigation control period	20 ms (50 Hz)
Δ_h, K	Prediction step and horizon	50 ms, 20 steps (1.0 s)
K_h	Handoff horizon	10 steps (0.5 s)
J, N_b	Control-aligned directions, height bands	21, 3
n_h	Consecutive mild violations before handoff	2 control steps
v_d	Design speed for scheduling and weights	4.5 m/s
δ, τ_v, a_b	Lag and braking parameters	Identified on hardware
$\bar{v}_c(h, \rho)$	Negotiation speed cap	Lookup table from Isaac Lab study

where v_k^{qp} is the predicted speed profile of the current pre-shaping solution (or of the reference in cruise mode) and $K_h \leq K$ is a shorter handoff horizon over which predictions are most reliable. Control is handed to the braking policy when any of the following holds:

$$h_t = \mathbb{K} \left[\Phi_t < -\varepsilon_h \vee \bigwedge_{i=0}^{n_h-1} \{\Phi_{t-i} < 0\} \vee \text{QP infeasible} \vee \dot{r}_t < -(v_t + \bar{v}_{\text{obs}}) \right], \quad (14)$$

where $\varepsilon_h > 0$ is a deep-violation threshold that triggers immediately, n_h is the number of consecutive mild violations required otherwise, $\dot{r}_t$ is the finite-difference rate of the present range $\min_b r_t(j_t^*, b)$, and $\bar{v}_{\text{obs}}$ is the largest obstacle speed for which a pre-shaped response is attempted. The last condition detects closure faster than the robot’s own motion explains, which indicates an obstacle moving toward the robot. The confirmation count n_h suppresses spurious handoffs caused by single-frame prediction noise, and the deep-violation threshold bounds the delay that this filtering can introduce. Control returns to pre-shaping when $\Phi_t > \phi_r$ for a release time T_r , with $\phi_r > 0$ providing hysteresis against chattering between modes.

The braking policy is a separate locomotion policy trained in Isaac Lab with PPO [43] to bring the robot from running speed to a stable stance as quickly as friction allows while keeping the base level and the heading fixed. It is trained with domain randomization of friction, payload, and command latency [44]. Separating it from the velocity-tracking policy avoids asking a smooth command-tracking policy to execute an abrupt stop it was not trained for.

4.6 Implementation

Table 1 lists the controller configuration. All parameters except the identified lag parameters ($\delta, \tau_v, \delta_\omega, \tau_\omega$), the braking deceleration a_b , and the negotiation speed cap table $\bar{v}_c(h, \rho)$ are shared across classes. The lag and braking parameters are identified on the hardware; the speed cap table and the remaining weights are selected in the Isaac Lab design-space study over five seeds. The design speed at which the lead-time schedule and weights are tuned is $v_d = 4.5$ m/s unless stated otherwise; Section 5 reports a sweep over v_d .

05 Experiments

5.1 Platform and course

All hardware trials use a Unitree Go2 on a controlled course consisting of a laboratory section and an indoor track. Six ob-

stacle classes are evaluated: low beam, mid post, high beam, moving cart, static clutter, and occluded gap. Approach speeds range from 4.0 to 6.0 m/s in 0.5 m/s steps, giving five speeds, and each class is run 20 times at each speed. Each method is therefore evaluated on $6 \times 5 \times 20 = 600$ trials. The design space of the controller, including the negotiation speed cap table and the controller weights, was explored in Isaac Lab with five seeds before the hardware campaign; the hardware trials constitute the reported results.

PROTOCOL NOTE

The approach speed is the commanded speed on the run-up before any obstacle-induced command change. All methods share the same locomotion policy, the same FutureRay module, and the same course configuration for each class; only the navigation layer differs.

5.2 Baselines and metrics

Three baselines are compared with the proposed controller. *Present-range reactive only* uses the same direction selection and stopping envelope but evaluates them on the present range r_t and reduces the command only once $m_t(0) < 0$, with an emergency stop on deep violation. *Present-range + speed limit* adds a global cap on the commanded speed to the reactive baseline, trading speed for lag distance. *Predicted range, no pre-shaping* replaces the present range with the predicted field in the reactive rule, so that it reacts to predicted conflicts but uses neither the lead-time schedule nor the pre-shaping program; its stop is triggered by the same reactive rule.

Four metrics are reported: the mean over classes of the maximum safe speed (5); the pooled miss rate; the pre-shape lead time, defined as the interval between the first obstacle-induced change of the velocity command and the arrival of the robot’s front at the obstacle’s conflict boundary, averaged over trials; and the braking handoff success, defined as the percentage of trials with a stop event (a braking handoff or, for the baselines, an emergency stop) in which the robot came to rest or resumed the negotiation without contact.

5.3 Main comparison

Table 2 summarizes the comparison. The proposed controller attains the highest mean maximum safe speed, 4.90 m/s, which is 0.18 m/s above the present-range reactive baseline (4.72 m/s). It also attains the lowest miss rate, 3.8% against 4.2%, a reduction of 0.4 percentage points, and the highest braking handoff success, 97.2% against 90.2%, an increase of 7.0 points. The pre-shape lead time is 0.31 s against 0.29 s.

Three observations follow from the baselines. First, the global speed limit lowers the maximum safe speed to 4.52 m/s, 0.20 m/s below the unlimited reactive baseline, without changing the miss rate (4.2% in both cases), although it raises handoff success from 90.2% to 93.3%. Limiting speed everywhere therefore makes stops more reliable but does not prevent the misses that occur, which suggests that these misses are caused by when the command changes rather than by the cruising speed alone. Second, feeding the predicted field into a reactive rule does not help: the predicted-range baseline is slightly worse than the present-range baseline on every metric, with 4.65 m/s, a 4.3% miss rate, the shortest lead time (0.26 s), and the lowest handoff success (89.1%). In this study,

Table 2: Main comparison on the Unitree Go2 course (six obstacle classes, 4.0–6.0 m/s, 20 trials per class and speed).

Method	Max safe speed, mean over classes (m/s)	Miss rate (%)	Pre-shape lead time (s)	Braking handoff success (%)
Present-range reactive only	4.72	4.2	0.29	90.2
Present-range + speed limit	4.52	4.2	0.30	93.3
Predicted range, no pre-shaping	4.65	4.3	0.26	89.1
Ours	4.90	3.8	0.31	97.2

Table 3: Ablation of the proposed controller.

Variant	Max safe speed, mean over classes (m/s)	Braking handoff success (%)
Ours (pre-shaping + handoff)	4.90	97.2
w/o braking handoff rule	4.34	81.4
w/o predicted range (present only)	4.12	75.5
w/o lead-time scheduling	4.34	93.1

the predicted field was useful only when the controller was built around it. Third, the difference in lead time between the proposed controller and the reactive baseline is small, 0.02 s, while the gap to the predicted-range baseline is 0.05 s. The gains of the proposed controller should therefore be attributed to the combination of when and how the command is shaped and to the handoff rule, not to a large shift in the onset of the response.

Per-class behavior of the reactive baseline illustrates the role of lag. On mid posts its maximum safe speed tops out at 4.1 m/s, 0.62 m/s below its own mean over classes. A mid post is fully visible in the present range well before the conflict, so this limit is not set by perception; it is set by the lag distance that the reactive command spends after the conflict is detected.

5.4 Ablation

Table 3 removes one component at a time. Without the braking handoff rule, the pre-shaping layer remains in control when the predicted range collapses, and stop events are handled by the reactive emergency stop; the mean maximum safe speed falls by 0.56 m/s to 4.34 m/s, and the success of stop events falls by 15.8 points to 81.4%. Without lead-time scheduling, pre-shaping is triggered at a fixed lead time; the maximum safe speed falls to the same 4.34 m/s, but stop events remain comparatively reliable at 93.1%, 4.1 points below the full controller. The two components therefore act on different parts of the problem: scheduling primarily determines the achievable speed, whereas the handoff rule primarily determines whether stops succeed.

Replacing the predicted field with the present range inside the full controller produces the largest degradation: 4.12 m/s, 0.78 m/s below the full controller, and 75.5% handoff success, 21.7 points below. This variant is also 0.60 m/s below the present-range reactive baseline of Table 2. A plausible explanation is that the lead-time schedule, the trigger (10), and the handoff margin (13) were tuned assuming a forward-looking margin. Fed with present range, the controller detects conflicts no earlier than the reactive baseline but applies a schedule that assumes earlier detection, and the handoff fires in states where a stop is no longer reliably feasible. This interpretation is consistent with the low handoff success but has not been tested directly.

Table 4: Sweep over the design (setpoint) speed of the controller.

Setpoint speed (m/s)	Max safe speed, mean over classes (m/s)	Miss rate (%)
4.0	4.92	2.9
4.5	4.9	3.8
5.0	4.83	5.4
5.5	4.61	8.9
6.0	4.2	14.2

5.5 Design-speed sweep

Table 4 varies the design speed v_d (the setpoint at which the lead-time schedule and controller weights are tuned), with the hardware evaluation grid unchanged. The row at 4.5 m/s corresponds to the configuration of Tables 2 and 3.

Performance degrades slowly up to 5.0 m/s and steeply beyond it. From 4.0 to 5.0 m/s the mean maximum safe speed drops by 0.09 m/s and the miss rate rises by 2.5 points; from 5.0 to 6.0 m/s the maximum safe speed drops by 0.63 m/s and the miss rate rises by 8.8 points, to 14.2%. Tuning for a high design speed makes the schedule less conservative at every approach speed, so pre-shaping begins later and misses accumulate at the faster approach speeds. The 4.0 m/s setting is marginally better than the 4.5 m/s setting used in the main comparison, by 0.02 m/s in maximum safe speed and 0.9 points in miss rate. The main comparison therefore does not use the best configuration found in the sweep, although this difference is small compared with the degradation above 5.0 m/s.

5.6 Failure cases

Occluded gap at 6.0 m/s. The occluded gap class reaches a miss rate of 22% at 6.0 m/s. In these trials the free range behind the occluder is not observable until the occluder is passed, and the predicted field cannot recover information that is absent from the history. The binding constraint is the range field behind the occluder, not the controller: the pre-shaping program and the handoff rule respond once the field changes, but too little range remains at that speed. Raising the safe speed for this class requires either a different viewpoint or a prior on the geometry behind occluders, not a different control law.

Late handoff with the moving cart at 5.5 m/s. In one trial with the moving cart at 5.5 m/s, the handoff fired 0.08 s late because of a 38 ms latency spike in the sensing-to-command pipeline. The cart passed 9 cm clear, so the trial was not a miss, but it reveals the sensitivity of the handoff to latency. At 5.5 m/s, 0.08 s corresponds to 0.44 m of travel and to four navigation control periods of 20 ms. The confirmation count n_h adds part of this delay by design, and a latency spike lengthens the time over which the confirmations accumulate. A natural remedy is to bypass the confirmation when the closure-rate condition of (14) is active, or to make n_h depend on measured latency.

Lag-limited reactive baseline on mid posts. As noted above, the present-range baseline’s maximum safe speed tops out at 4.1 m/s on mid posts. Because the mid post is a fully visible,

static obstacle, this case isolates controller lag as the limit: the obstacle is perceived in time, but the reactive command cannot change the body's speed before the conflict.

06 Limitations

The evaluation uses one platform, one locomotion policy, and one controlled course with six obstacle classes. The results do not establish performance outdoors, on uneven or low-friction ground, under varying lighting, or with obstacle types outside the six classes, and the moving-obstacle class consists of a single cart. Each cell of the evaluation grid contains 20 trials, and confidence intervals are not reported; the differences in miss rate between methods, for example 0.4 points between the proposed controller and the reactive baseline, are small relative to what 20 trials per cell can resolve for individual classes and should be read as aggregate trends. The Isaac Lab design-space study was used to set controller parameters, including the negotiation speed cap as a function of obstacle height and field-of-view range; its per-height and per-range results are not reported in this paper, and the agreement of that table with hardware behavior has not been quantified separately.

The controller relies on the predicted range field and its uncertainty estimate. Prediction errors were not measured independently of control outcomes, so the contribution of perception error to individual misses cannot be separated. The lag model is a delay plus a first-order response; gait-dependent or speed-dependent lag is not modeled, and the latency spike in the moving-cart case shows that unmodeled latency can delay the handoff. The stopping-envelope surrogate is conservative at low speed, and the handoff rule carries no formal safety guarantee. Finally, the occluded-gap failure is a limit of observability that the present design does not address.

07 Conclusion

This paper studied high-speed obstacle negotiation for quadrupeds when the reaction budget is dominated by velocity-controller lag. Building on the control-aligned future range predicted by FutureRay, the proposed controller pre-shapes the forward and yaw commands with a lead time scheduled from an identified lag model, enforces a conservative stopping envelope against the predicted range, and hands control to a dedicated braking policy when the predicted margin collapses. On a Unitree Go2 over six obstacle classes at 4–6 m/s, it raised the mean maximum safe speed from 4.72 to 4.90 m/s, lowered the miss rate from 4.2% to 3.8%, and raised braking handoff success from 90.2% to 97.2% relative to a present-range reactive baseline. The baselines and ablation indicate that the predicted range is beneficial only when the controller is designed around it, and that scheduling and handoff mainly address speed and stop reliability, respectively. The remaining failures point to observability behind occluders and latency robustness of the handoff as the next problems to address.

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] 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.
- [3] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning Quadrupedal Locomotion over Challenging Terrain. arXiv:2010.11251, 2020.
- [4] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning. arXiv:2109.11978, 2021.
- [5] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid Motor Adaptation for Legged Robots. arXiv:2107.04034, 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. arXiv:2201.08117, 2022.
- [7] 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.
- [8] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal. Rapid Locomotion via Reinforcement Learning. arXiv:2205.02824, 2022.
- [9] G. B. Margolis and P. Agrawal. Walk These Ways: Tuning Robot Control for Generalization with Multiplicity of Behavior. arXiv:2212.03238, 2022.
- [10] I. M. A. Nahrendra, B. Yu, and H. Myung. DreamWaQ: Learning Robust Quadrupedal Locomotion with Implicit Terrain Imagination via Deep Reinforcement Learning. arXiv:2301.10602, 2023.
- [11] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged Locomotion in Challenging Terrains using Egocentric Vision. arXiv:2211.07638, 2022.
- [12] R. Yang, G. Yang, and X. Wang. Neural Volumetric Memory for Visual Locomotion Control. arXiv:2304.01201, 2023.
- [13] D. Hoeller, N. Rudin, C. Choy, A. Anandkumar, and M. Hutter. Neural Scene Representation for Locomotion on Structured Terrain. arXiv:2206.08077, 2022.
- [14] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot Parkour Learning. arXiv:2309.05665, 2023.
- [15] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme Parkour with Legged Robots. arXiv:2309.14341, 2023.
- [16] D. Hoeller, N. Rudin, D. Sako, and M. Hutter. ANYmal Parkour: Learning Agile Navigation for Quadrupedal Robots. arXiv:2306.14874, 2023.
- [17] S. Ha, J. Lee, M. van de Panne, Z. Xie, W. Yu, and M. Khadiv. Learning-based Legged Locomotion: State of the Art and Future Perspectives. arXiv:2406.01152, 2024.
- [18] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter. Advanced Skills by Learning Locomotion and Local Navigation End-to-End. arXiv:2209.12827, 2022.
- [19] D. Hoeller, L. Wellhausen, F. Farshidian, and M. Hutter. Learning a State Representation and Navigation in Cluttered and Dynamic Environments. arXiv:2103.04351, 2021.
- [20] 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.
- [21] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi. Agile But Safe: Learning Collision-Free High-Speed Legged Locomotion. arXiv:2401.17583, 2024.
- [22] 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.
- [23] M. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter. Fast Traversability Estimation for Wild Visual Navigation. arXiv:2305.08510, 2023.

- [24] G. Kahn, P. Abbeel, and S. Levine. BADGR: An Autonomous Self-Supervised Learning-Based Navigation System. arXiv:2002.05700, 2020.
- [25] D. Fox, W. Burgard, and S. Thrun. The Dynamic Window Approach to Collision Avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33, 1997.
- [26] P. Fiorini and Z. Shiller. Motion Planning in Dynamic Environments Using Velocity Obstacles. *International Journal of Robotics Research*, 17(7):760–772, 1998.
- [27] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada. Control Barrier Function Based Quadratic Programs for Safety Critical Systems. arXiv:1609.06408, 2016.
- [28] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control Barrier Functions: Theory and Applications. arXiv:1903.11199, 2019.
- [29] K.-C. Hsu, H. Hu, and J. F. Fisac. The Safety Filter: A Unified View of Safety-Critical Control in Autonomous Systems. arXiv:2309.05837, 2023.
- [30] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-Jacobi Reachability: A Brief Overview and Recent Advances. arXiv:1709.07523, 2017.
- [31] A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani, and C. J. Tomlin. An Efficient Reachability-Based Framework for Provably Safe Autonomous Navigation in Unknown Environments. arXiv:1905.00532, 2019.
- [32] K. P. Wabersich and M. N. Zeilinger. A Predictive Safety Filter for Learning-Based Control of Constrained Nonlinear Dynamical Systems. arXiv:1812.05506, 2018.
- [33] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan. Bridging the Gap Between Safety and Real-Time Performance in Receding-Horizon Trajectory Design for Mobile Robots. arXiv:1809.06746, 2018.
- [34] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How. FASTER: Fast and Safe Trajectory Planner for Navigation in Unknown Environments. arXiv:2001.04420, 2020.
- [35] 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.
- [36] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. Learning High-Speed Flight in the Wild. arXiv:2110.05113, 2021.
- [37] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza. Champion-Level Drone Racing Using Deep Reinforcement Learning. *Nature*, 620:982–987, 2023.
- [38] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza. Reaching the Limit in Autonomous Racing: Optimal Control versus Reinforcement Learning. arXiv:2310.10943, 2023.
- [39] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim. Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.
- [40] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter. Perceptive Locomotion through Nonlinear Model Predictive Control. arXiv:2208.08373, 2022.
- [41] V. Makoviychuk et al.. Isaac Gym: High Performance GPU-Based Physics Simulation for Robot Learning. arXiv:2108.10470, 2021.
- [42] M. Mittal et al.. Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments. arXiv:2301.04195, 2023.
- [43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. arXiv:1707.06347, 2017.
- [44] 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.