

Latent Motion Tokens for Future-Range Prediction in Quadruped Obstacle Avoidance

Marcus Hill • Ivan Sokolov • Yating Feng • Rachel Kim

Robotics Institute, Georgia Institute of Technology, Atlanta, GA, USA

PUBLISHED

August 4, 2026

REGISTERED

August 6, 2026

RECORD TYPE

Preprint

ABSTRACT

Learned range predictors for legged-robot navigation usually regress the free distance along a fixed set of rays. This output format is difficult to condition on the motion the robot intends to execute, and it forces the planner and the reactive safety layer to share a single ray resolution. This paper encodes the predicted future range field of a quadruped as a small set of latent motion tokens, one per cluster of control-aligned directions. A transformer generates the tokens from recent range history and the planned motion, and a lightweight query decoder returns the future free range for any ray, horizon, and candidate motion. Because queries are resolved against the tokens rather than a fixed output layer, the same predictor serves a 2 Hz planner issuing coarse queries and a 50 Hz reactive layer issuing dense queries. In Isaac Lab experiments with ANYmal in 20 cluttered obstacle-avoidance scenes, with 5 seeds and 30 trials per scene and seed, the token predictor attains a range prediction error of 0.87 m, a 95th-percentile query latency of 13.6 ms on the onboard Orin computer, and an avoidance success rate of 91.8 percent, compared with 0.91 m, 14.4 ms, and 79.8 percent for a FutureRay-style direct-regression baseline, while using 1.2 million instead of 1.3 million parameters. Ablations indicate that motion-plan conditioning and the number of tokens matter most, and the recorded failure cases identify occluded turns and planner stalls as remaining weaknesses.

INDEX TERMS quadruped navigation, future range prediction, latent motion tokens, obstacle avoidance, query-based decoding, ANYmal

KAGG NOTE Representing the future range field as plan-conditioned latent tokens that are decoded on demand lets one predictor serve both slow, coarse planning queries and fast, dense reactive queries.

01 Introduction

Quadruped robots are now routinely deployed in cluttered industrial sites, construction areas, and buildings, where they must avoid obstacles while walking at speeds on the order of one meter per second. Learned locomotion controllers have made the low-level gait robust to rough terrain and disturbances [1, 2, 3, 4], so the limiting factor for safe navigation is increasingly the perception interface that tells the higher layers how much free space is available in the directions the robot is about to move. A range scan describes the scene at the current instant and in the current body frame. During agile motion, however, the robot will occupy a different pose by the time a command takes effect, and the relevant question is how much free range will be available along the motion the controller is actually going to execute.

FutureRay [5] addresses this question by predicting future free range along the robot's own predicted control-aligned directions, so that obstacle queries are aligned with motions the controller can execute. Its output, like that of most learned range predictors, is a direct per-ray regression: the network emits one free-distance value for each element of a fixed set of rays and horizons. This representation has two practical drawbacks. First, it is hard to condition on a motion plan that changes over time, because the plan enters only through the input while the output layer is tied to a fixed geometry. Second, the consumers of the prediction do not want the same

rays. A global or local planner that replans at a few hertz evaluates many candidate motions with a few coarse queries each, whereas a reactive safety layer that runs at the control rate needs dense queries around the current command. With a fixed output layer, one must either train separate predictors or interpolate a single fixed output for both consumers.

This paper proposes to represent the predicted future range field as a small set of *latent motion tokens*. Each token summarizes the future free range within one cluster of control-aligned directions. A transformer generates the tokens from recent range history and the planned motion; a lightweight decoder then answers arbitrary ray queries, each specified by a bearing, a future time offset, and an optional deviation from the nominal plan. Because the decoder is conditioned on the query rather than tied to a fixed output layout, the same token set serves a 2 Hz planner issuing coarse queries and a 50 Hz reactive layer issuing dense queries.

The contributions are as follows.

1. A latent-token representation of the control-aligned future range field, in which a fixed number of tokens, each associated with a direction cluster defined relative to the predicted motion, replaces the per-ray output layer of direct regression.
2. A plan-conditioned token generator and a query-conditioned decoder that resolve ray queries at arbitrary bearing, horizon, and plan deviation, together with a training scheme

that randomly drops the plan so that the predictor degrades to present-range behavior when the plan is unavailable.

3. A two-rate integration in which a 2 Hz sampling planner and a 50 Hz reactive speed-scaling filter query the same predictor at different densities, including an explicit fallback to present range under high predictive uncertainty or a stale plan.
4. An evaluation in Isaac Lab with the ANYmal quadruped across 20 cluttered scenes, 5 seeds, and 30 trials per scene and seed, with query latency measured on the onboard Orin computer, comparing against a FutureRay-style direct-regression baseline and a present-range predictor, together with ablations and an analysis of recorded failure cases.

Section 2 reviews related work, Section 3 formalizes the prediction problem, Section 4 develops the method, Section 5 reports the experiments, and Sections 6 and 7 discuss limitations and conclude.

02 Related Work

2.1 Learned locomotion and navigation for legged robots

Reinforcement learning in simulation has become the dominant way to obtain robust quadruped controllers. Hwangbo et al. [1] trained ANYmal policies in simulation with a learned actuator model and transferred them to hardware, and Lee et al. [2] used a teacher-student scheme to traverse challenging natural terrain blindly. Miki et al. [3] added exteroception through an attention-based belief encoder that fuses proprioception with a height map and learns when to distrust it. Rudin et al. [4] showed that massively parallel GPU simulation reduces training to minutes, a capability built on Isaac Gym [6] and later the Orbit framework [7] that underlies Isaac Lab, the simulator used in this study. Rapid motor adaptation [8] and vision-guided controllers [9, 10] further broadened the terrain these policies handle, and parkour systems [11, 12, 13] demonstrated agile motion that couples perception and locomotion tightly. These works mostly use perception to select footholds or body motion over terrain; the present paper instead assumes a velocity-tracking locomotion policy and focuses on the free-space prediction that drives the velocity commands themselves.

Several works learn navigation on top of such controllers. Rudin et al. [14] trained locomotion and local navigation end-to-end, and Hoeller et al. [15] learned a compact state representation from depth data on which a navigation policy operates in cluttered and dynamic environments. Truong et al. [16] learned robot embeddings that transfer navigation skills to legged robots, and Kim et al. [17] learned a forward dynamics model together with an informed trajectory sampler for safe quadruped navigation. Frey et al. [18] estimated traversability online from self-supervision. Perceptive model-predictive control [19] offers a model-based alternative in which the terrain representation enters the optimization directly. The present work is complementary: it provides a prediction interface that both a sampling planner and a reactive filter can consume, rather than a policy that maps observations to actions.

2.2 Mapless navigation from range data

End-to-end mapless navigation from range sensors maps a scan directly to motion commands. Pfeiffer et al. [20] imitated a global planner from 2-D laser scans, Tai et al. [21] trained a

mapless navigation policy with deep reinforcement learning in simulation and transferred it to a real robot, and Chiang et al. [22] used AutoRL to tune rewards and architectures for point-to-point and path-following navigation from lidar. Kahn et al. [23] learned predictive models of collision events conditioned on action sequences, and BADGR [24] extended this to self-supervised event prediction from real-world experience. The action-conditioned prediction in these models is close in spirit to the plan conditioning used here, but their outputs are task events such as collisions rather than a geometric range field that can be queried at arbitrary bearings. Generalist visual navigation models such as GNM [25], ViNT [26], and NoMaD [27] learn goal-directed behavior from heterogeneous data but rely on a separate safety mechanism for near-field avoidance, which is the role of the reactive layer in this paper. High-speed flight [28] shows that learned perception-to-trajectory maps can handle clutter at speed when the training data covers the relevant motions.

2.3 Predicting future occupancy and range

A substantial literature forecasts the future state of the environment. Mohajerin and Rohani [29] predicted multi-step occupancy grids with recurrent networks, and Itkina et al. [30] learned recurrent representations for dynamic occupancy prediction in urban scenes. Weng et al. [31] forecast entire future point clouds before detection, and Mersch et al. [32] used 3-D spatio-temporal convolutions to predict future range images in a self-supervised manner. ChauffeurNet [33] conditioned driving predictions on rendered intended routes. These methods forecast scene geometry in a world or sensor frame, which is appropriate for dynamic environments observed by a static or slowly moving platform. For a quadruped executing an aggressive plan, most of the change in the range observed along a given direction comes from the robot’s own motion, and the relevant frame is the predicted future body frame.

2.4 Control-aligned future range

FutureRay [5] is the work most closely related to this paper. It provides a perception module that predicts future free range along the robot’s own predicted control-aligned directions, so that obstacle queries are aligned with what the controller can actually execute. Concretely, the prediction is expressed relative to the robot’s predicted future motion rather than its current heading, which removes a mismatch between where a present-time scan looks and where the robot will move. The prediction is produced by direct regression of per-ray free distances over a fixed set of directions and horizons. This paper adopts FutureRay’s control-aligned frame and its target quantity, future free range, and differs in the representation and the interface. Instead of a per-ray output layer, the future range field is summarized by a fixed set of latent tokens indexed by direction clusters; the motion plan conditions token generation; and the decoder answers queries whose bearing, horizon, and plan deviation are inputs rather than output indices. This allows one predictor to serve consumers with different query densities and allows candidate motions that deviate from the nominal plan to be evaluated without re-running the full network. The experiments compare against a FutureRay-style direct-regression predictor implemented in the same stack with a similar parameter budget; this is a re-implementation of the representation and not an evaluation of the original system.

2.5 Latent tokens and query-based decoders

The architecture draws on several lines of work in representation learning. The transformer [34] provides attention-based sequence modeling. Set Transformer [35] and Perceiver [36] compress large inputs into a small set of latent vectors through cross-attention with learned inducing points or latent arrays, which bounds computation independently of the input size; Perceiver IO [37] adds query-based output decoding, in which each output is obtained by cross-attending from a query vector to the latents. DETR [38] uses learned object queries to decode a set prediction. Implicit neural representations such as occupancy networks [39], DeepSDF [40], and NeRF [41] decode continuous geometric fields at arbitrary query coordinates, and Fourier feature embeddings [42] enable such decoders to represent high-frequency structure. Latent world models [43] show that compact latent states can support prediction and control. The motion tokens in this paper combine these ideas: a Perceiver-style bottleneck with tokens tied to control-aligned direction clusters, and a query decoder in the style of implicit fields, with queries that include motion deviations.

2.6 Reactive safety layers

Layered architectures in which a slow planner proposes motions and a fast layer enforces safety are standard in mobile robotics. Control barrier functions [44] formalize such a layer as a minimally invasive filter that keeps the state within a safe set. The reactive layer in this paper is a simpler speed-scaling filter based on a stopping-distance condition, but it shares the principle that safety enforcement runs at the control rate and consumes a conservative estimate of free space. Predictive uncertainty enters this estimate through a heteroscedastic output [45], which provides both a lower confidence bound on free range and a trigger for fallback to present range.

03 Problem Formulation

3.1 Platform, sensing, and commands

The robot is an ANYmal quadruped driven by a learned locomotion policy that tracks planar body-velocity commands $\mathbf{u} = (v_x, v_y, \omega)$, where v_x and v_y are the forward and lateral velocities in the body frame and ω is the yaw rate. Commands are issued at the control period $\Delta_c = 20$ ms, corresponding to 50 Hz. The planar pose at time t is $\mathbf{x}_t = (p_t, \psi_t) \in SE(2)$ with position $p_t \in \mathbb{R}^2$ and heading ψ_t .

Exteroception is summarized as a polar range scan. LiDAR returns within a height band that intersects the robot body are projected onto the horizontal plane and binned into N_r bearings $\{\beta_j\}_{j=1}^{N_r}$ uniformly covering $[-\pi, \pi)$. The scan $\mathbf{r}_t \in [0, r_{\max}]^{N_r}$ stores the minimum range per bin, clipped at $r_{\max}$. The range history is $\mathcal{R}_t = (\mathbf{r}_{t-(L-1)\Delta_s}, \dots, \mathbf{r}_t)$ with scan period Δ_s ; older scans are re-expressed in the current body frame using odometry.

3.2 Motion plan and control-aligned frame

A motion plan is a sequence of H commands over the horizon $T_h = H\Delta_p$,

$$\pi_t = (\mathbf{u}_t, \mathbf{u}_{t+\Delta_p}, \dots, \mathbf{u}_{t+(H-1)\Delta_p}). \quad (1)$$

Integrating the plan from the current pose gives the predicted pose $\hat{\mathbf{x}}_{t+\tau} = (\hat{p}_{t+\tau}, \hat{\psi}_{t+\tau}) = \Phi(\mathbf{x}_t, \pi_t, \tau)$ for $\tau \in [0, T_h]$, where Φ applies the commanded body velocities on $SE(2)$. Following FutureRay [5], directions are expressed relative to

the predicted direction of travel. The control-aligned axis at offset τ is

$$\alpha_{t+\tau} = \hat{\psi}_{t+\tau} + \text{atan2}(\hat{v}_{y,t+\tau}, \hat{v}_{x,t+\tau}), \quad (2)$$

where $(\hat{v}_x, \hat{v}_y)$ is the commanded translational velocity at that time. When the commanded translational speed is below a threshold v_e , the axis is defined as the predicted heading $\hat{\psi}_{t+\tau}$.

3.3 Future range field and queries

Let $\mathcal{O} \subset \mathbb{R}^2$ denote the occupied set at the height band of the body. For a plan π , the future range field is

$$\begin{aligned} \rho_t(\theta, \tau | \pi) &= \min \{r_{\max}, \inf S_t(\theta, \tau | \pi)\}, \\ S_t(\theta, \tau | \pi) &= \{s \geq 0 : \hat{p}_{t+\tau} + s \mathbf{e}(\alpha_{t+\tau} + \theta) \in \mathcal{O}\}, \end{aligned} \quad (3)$$

where $\mathbf{e}(\cdot)$ is the unit vector at the given angle and $\theta \in [-\pi, \pi)$ is a bearing relative to the control-aligned axis. The field therefore measures the free distance from the predicted future position along directions defined by the predicted future motion.

Consumers interact with the predictor through queries. A query is a tuple $q = (\theta, \tau, \delta)$, where $\delta = (\delta_v, \delta_\omega)$ is a deviation from the nominal plan: a multiplicative speed scale offset δ_v and an additive yaw-rate offset δ_ω applied to every command of π_t . The deviated plan is denoted $\pi_t \oplus \delta$, and the query target is $\rho_t(\theta, \tau | \pi_t \oplus \delta)$. A query with $\delta = \mathbf{0}$ refers to the nominal plan.

3.4 Prediction problem

The goal is a predictor f that, given the history and nominal plan, produces for any query an estimate and an uncertainty,

$$(\hat{\rho}_t(q), \hat{\sigma}_t(q)) = f(q; \mathcal{R}_t, \pi_t), \quad (4)$$

such that the error $|\hat{\rho}_t(q) - \rho_t(q)|$ is small over the query distributions of both consumers and the per-query cost is low enough for a 50 Hz consumer. The two consumers differ in query distribution: the planner issues few queries per candidate across many candidates and long horizons, whereas the reactive layer issues many queries at short horizons near the current command.

04 Method

4.1 Overview

The predictor factors f into a token generator and a query decoder,

$$Z_t = G(\mathcal{R}_t, \pi_t), \quad (\hat{\rho}_t(q), \hat{\sigma}_t(q)) = D(q, Z_t), \quad (5)$$

where $Z_t \in \mathbb{R}^{K \times d}$ contains K motion tokens of width d . The generator runs when a new scan or a new plan arrives; the decoder runs per query batch. The design rests on the observation that free range along neighboring control-aligned directions is strongly correlated, so a small number of tokens can carry the information that a direct regressor spreads over hundreds of output units, while the decoder recovers fine angular structure from the query embedding. The default configuration uses $K = 16$ tokens; Section 5.5 reports a variant with 4 tokens.

4.2 Direction clusters

Each token is associated with a cluster of control-aligned bearings. The bearing circle relative to the control-aligned axis is partitioned into K contiguous sectors $\mathcal{S}_k = [\theta_k^-, \theta_k^+)$ with centers $\bar{\theta}_k$. Sectors are narrower near the direction of travel, where

obstacles affect the next commands most, and wider behind the robot. The partition is obtained by uniform spacing in a warped coordinate,

$$\theta_k^\pm = \text{sign}(\xi_k^\pm) \pi |\xi_k^\pm|^\gamma, \quad \xi_k^\pm \in \{-1, -1 + \frac{2}{K}, \dots, 1\}, \quad (6)$$

with exponent $\gamma > 1$, so that sectors shrink toward $\theta = 0$. Because the sectors are defined in the control-aligned frame of Eq. (2), a token keeps its meaning as the plan turns: the token whose sector contains $\theta = 0$ always describes the region ahead along the predicted motion.

4.3 Input tokenization

Each scan in $\mathcal{R}_t$ is split into P contiguous bearing patches of N_r/P bins. A patch is embedded by a linear map with added learned embeddings for the patch bearing and the scan age, giving LP scan tokens. Each plan command $\mathbf{u}_{t+h\Delta_p}$ is embedded by a two-layer perceptron together with a time embedding and with the predicted pose increment $\hat{\mathbf{x}}_{t+h\Delta_p} \ominus \mathbf{x}_t$, giving H plan tokens. The input set is $X_t = [X_t^{\text{scan}}; X_t^{\text{plan}}] \in \mathbb{R}^{(LP+H) \times d}$. Including the integrated pose increments, rather than raw commands alone, gives the generator direct access to where the robot will be, which is the dominant source of change in the future range.

4.4 Motion-token generator

The tokens are initialized from learned per-sector embeddings $\mathbf{e}_k$ combined with a plan-dependent embedding of the sector geometry,

$$\mathbf{z}_k^{(0)} = \mathbf{e}_k + \text{MLP}_0(\bar{\theta}_k, \alpha_{t+\tau_1} - \psi_t, \dots, \alpha_{t+\tau_M} - \psi_t), \quad (7)$$

where $\tau_1, \dots, \tau_M$ are fixed reference offsets in $[0, T_h]$, so that each token knows how its sector sweeps through the current body frame as the plan unfolds. The generator then applies N_g blocks, each consisting of cross-attention from the tokens to the inputs, self-attention among the tokens, and a feed-forward layer, all with pre-normalization and residual connections:

$$Z' = Z^{(n)} + \text{CA}(\text{LN}(Z^{(n)}), \text{LN}(X_t)), \quad (8)$$

$$Z'' = Z' + \text{SA}(\text{LN}(Z')), \quad (9)$$

$$Z^{(n+1)} = Z'' + \text{FFN}(\text{LN}(Z'')). \quad (10)$$

The cross-attention in Eq. (8) uses an additive geometric bias so that token k initially favors scan patches whose bearings intersect the region swept by sector k along the predicted path,

$$\text{logit}_{k,i} = \frac{\mathbf{q}_k^\top \mathbf{k}_i}{\sqrt{d_h}} - \lambda \min_m \text{dist}_\angle(b_i, \alpha_{t+\tau_m} - \psi_t + \bar{\theta}_k)^2, \quad (11)$$

where b_i is the bearing of scan patch i in the current body frame, $\text{dist}_\angle$ is the wrapped angular distance, d_h is the head dimension, and $\lambda \geq 0$ is learned per head; the bias is set to zero for plan tokens. The cost of the generator scales as $O(K(LP+H)d)$ for cross-attention and $O(K^2d)$ for self-attention, which is linear in the input size and independent of the number of queries.

Plan dropout. During training, the plan tokens are replaced with probability p_{drop} by a learned no-plan embedding, and the target is computed for a stationary plan, so that $\hat{\mathbf{x}}_{t+\tau} = \mathbf{x}_t$ and the field reduces to the present range in the current heading frame. At run time, the same substitution is applied when the

plan is missing or stale. The predictor thus degrades by design to a present-range estimator rather than extrapolating with an outdated plan.

4.5 Query decoder

A query $q = (\theta, \tau, \delta)$ is embedded as

$$\mathbf{h}_q = \text{MLP}_q(\gamma(\theta), \gamma(\tau/T_h), \delta), \quad (12)$$

with the Fourier embedding [42]

$$\gamma(x) = [\sin(2^\ell \pi x), \cos(2^\ell \pi x)]_{\ell=0}^{F-1} \quad (13)$$

applied to θ/π and τ/T_h . The decoder cross-attends from $\mathbf{h}_q$ to the tokens with a sector-locality bias,

$$\begin{aligned} \mathbf{c}_q &= \sum_{k=1}^K a_{qk} W_v \mathbf{z}_k, \\ a_{qk} &= \text{softmax}_k \left(\frac{(W_q \mathbf{h}_q)^\top W_k \mathbf{z}_k}{\sqrt{d}} - \mu \text{dist}_\angle(\theta, \bar{\theta}_k)^2 \right), \end{aligned} \quad (14)$$

and a two-layer head maps $[\mathbf{h}_q; \mathbf{c}_q]$ to (o_ρ, o_σ) , from which

$$\hat{\rho}_t(q) = r_{\max} \text{sigmoid}(o_\rho), \quad \hat{\sigma}_t(q) = \sigma_{\min} + \text{softplus}(o_\sigma). \quad (15)$$

Queries are independent given Z_t , so a batch of queries is decoded with one batched matrix product per layer. The deviation δ enters only through the query; the tokens are computed once for the nominal plan. This is an approximation that holds for small deviations, which is the regime the planner explores around its previous solution, and it is what allows many candidates to be scored without regenerating tokens.

4.6 Training objective

Training samples consist of a history, a nominal plan, and a set of queries with ground-truth targets obtained by ray casting against the scene geometry from the integrated future pose of Eq. (3). Queries are drawn from a mixture that imitates both consumers: dense bearings at short horizons with $\delta = \mathbf{0}$, coarse bearings at all horizons with random deviations, and uniformly random queries. The loss is a weighted Laplace negative log-likelihood,

$$\mathcal{L} = \frac{1}{|Q|} \sum_{q \in Q} w(\rho_q) \left(\frac{|\rho_q - \hat{\rho}_q|}{\hat{\sigma}_q} + \log \hat{\sigma}_q \right), \quad (16)$$

where $w(\rho) = 1 + \eta \mathbb{I}[\rho < r_{\text{near}}]$ up-weights short ranges, because errors near obstacles matter more for avoidance than errors in open space. The Laplace likelihood matches the evaluation metric, mean absolute error, and provides the uncertainty used by both consumers [45]. The predictor was trained on 40 h of teleoperated and autonomous rollouts.

4.7 Planner: coarse queries at 2 Hz

The planner runs at 2 Hz. It forms N_c candidate plans $\pi_t \oplus \delta_m$ from a lattice of speed-scale and yaw-rate offsets around the previous plan shifted forward in time, and for each candidate issues coarse queries at N_θ^c bearings around the control-aligned axis and N_τ^c horizons. With the lower confidence bound

$$\underline{\rho}(q) = \hat{\rho}_t(q) - \kappa \hat{\sigma}_t(q), \quad (17)$$

each candidate is scored by

$$J(\delta_m) = w_g \|\hat{p}_{t+T_h}(\delta_m) - p_{\text{goal}}\| + w_c \sum_{\theta, \tau} \text{softplus}\left(\frac{d_{\text{safe}}(\theta) - \underline{\rho}(\theta, \tau, \delta_m)}{s}\right) + w_\delta \|\delta_m\|^2, \quad (18)$$

where $d_{\text{safe}}(\theta)$ is the distance from the body center to the footprint boundary along bearing θ plus a margin, and s sets the softness of the clearance penalty. The minimizing candidate becomes the new nominal plan, which conditions the next token generation. Because the same tokens serve all candidates, one replanning cycle requires one generator pass and $N_c N_\theta^c N_\tau^c$ decoder queries.

4.8 Reactive layer: dense queries at 50 Hz

The reactive layer runs at 50 Hz. It takes the current command $\mathbf{u}_t$ of the nominal plan and issues dense queries at N_θ^d bearings within the sector of the commanded motion and at short horizons $\tau \in [0, \tau_r]$. The admissible speed is the largest speed for which the robot can stop within the conservative clearance, accounting for a reaction delay t_d and a braking deceleration a_b :

$$v t_d + \frac{v^2}{2a_b} \leq \min_{\theta, \tau} [\underline{\rho}(\theta, \tau, \mathbf{0}) - d_{\text{safe}}(\theta)] =: c_t, \quad (19)$$

which gives

$$v_{\text{adm}} = \begin{cases} a_b \left(\sqrt{t_d^2 + 2c_t/a_b} - t_d \right), & c_t > 0, \\ 0, & \text{otherwise.} \end{cases} \quad (20)$$

The command sent to the locomotion policy is

$$\mathbf{u}'_t = \left(\min \left\{ 1, \frac{v_{\text{adm}}}{\|(v_x, v_y)\|} \right\} (v_x, v_y), \omega + k_\omega \Delta c_t \right), \quad (21)$$

where Δc_t is the difference between the conservative clearances of the left and right halves of the queried sector, which steers away from the tighter side. The minimum in Eq. (19) is taken over a bearing set wide enough to contain the body footprint swept along the command.

Fallback to present range. The reactive layer switches to the present scan, using $c_t = \min_\theta [\mathbf{r}_t(\alpha_t + \theta) - d_{\text{safe}}(\theta)]$ over the same bearing set, when any of three conditions holds: the mean predicted uncertainty $\hat{\sigma}$ over the dense queries exceeds a threshold σ_{max} , the plan is older than a staleness limit, or a fresh prediction is unavailable at the control deadline. In the last case the layer re-samples the most recent token set at the current time offset if it is younger than one scan period and logs the missed deadline.

4.9 Run-time scheduling

Token generation is triggered by each new scan and each new plan. Between generations, queries are adjusted for elapsed time: a query issued at time t' against tokens generated at t uses $\tau + (t' - t)$ as horizon and rotates bearings by the odometry yaw change, so that the decoder always answers in the frame of Eq. (3). The reported query latency is measured from the issue of a reactive query batch to the return of its estimates and includes a token regeneration when one falls in the same cycle.

The 20 ms control period is the latency budget of the reactive consumer.

IMPLEMENTATION NOTE

The token generator and decoder together have 1.2 million parameters in the configuration with $K = 16$ tokens. All latencies reported in Section 5 are measured on the onboard Orin computer of the robot.

05 Experiments

5.1 Setup

All closed-loop experiments are conducted in Isaac Lab with the ANYmal quadruped in 20 cluttered obstacle-avoidance scenes. Each scene is run with 5 seeds and 30 trials per seed, giving 150 trials per scene and 3000 trials per method. The reactive layer queries the predictor at 50 Hz and the planner at 2 Hz, as described in Section 4. The learned predictor is trained on 40 h of teleoperated and autonomous rollouts. Query latency is measured on the onboard Orin.

5.2 Compared predictors

All predictors are used within the same planner and reactive layer, so differences in avoidance success reflect the prediction interface.

- **FutureRay-style direct regression.** A re-implementation of the FutureRay representation [5]: a network that regresses future free range along control-aligned directions directly at a fixed set of rays and horizons. Queries that fall between output rays, horizons, or off the nominal plan are answered by interpolation of the fixed outputs.
- **Present-range MLP (no future).** A multilayer perceptron that estimates the present range field from the scan history and uses it as the estimate for all future horizons. It has no access to the plan.
- **Token predictor, no query conditioning.** The motion-token generator with a decoder that receives only the bearing of a query. The horizon and plan deviation are not part of the query, so all queries are answered for the nominal plan at a fixed horizon.
- **Ours.** The full method with $K = 16$ motion tokens, plan conditioning, and query-conditioned decoding.

5.3 Metrics

Range prediction error is the mean absolute error in meters between predicted and ground-truth future free range, Eq. (3), over evaluation queries. *Query latency p95* is the 95th percentile of the reactive query latency defined above. *Avoidance success* is the percentage of trials in which the robot reaches its goal without collision. *Predictor parameters* counts the learned parameters of the range predictor in millions.

5.4 Main results

Table 1 reports the main comparison.

The full token predictor achieves the lowest range prediction error (0.87 m), the lowest 95th-percentile query latency (13.6 ms), and the highest avoidance success (91.8%) with the smallest parameter count (1.2 M). Relative to the FutureRay-style baseline, the error decreases by 0.04 m, the p95 latency

Table 1: Main comparison in Isaac Lab with ANYmal (20 scenes, 5 seeds, 30 trials each). Latency is measured on the onboard Orin.

Method	Range prediction error (m)	Query latency p95 (ms)	Avoidance success (%)	Predictor parameters (M)
FutureRay-style direct regression	0.91	14.4	79.8	1.3
Present-range MLP (no future)	0.91	14.1	85.8	1.3
Token predictor, no query conditioning	0.95	15.5	79.2	1.4
Ours	0.87	13.6	91.8	1.2

Table 2: Ablation of the token predictor in the same evaluation protocol.

Variant	Range prediction error (m)	Avoidance success (%)
Ours (motion tokens)	0.87	91.8
w/o motion-plan conditioning	0.93	85.6
w/o token decoder (direct head)	0.96	88.3
4 tokens instead of 16	1.03	78.1

by 0.8 ms, and avoidance success increases by 12.0 percentage points. All four predictors stay within the 20 ms budget of the 50 Hz consumer at the 95th percentile.

The improvement in prediction error is modest compared with the improvement in avoidance success, and two observations indicate that aggregate error alone does not determine closed-loop behavior. First, the FutureRay-style baseline and the present-range MLP have identical error (0.91 m), yet the present-range MLP succeeds in 85.8% of trials against 79.8% for the baseline. Second, the token predictor without query conditioning has a lower success rate (79.2%) than either. A plausible explanation, which these experiments do not isolate, is that both the baseline and the unconditioned token predictor answer off-grid and off-plan queries by interpolation or by reusing nominal-plan predictions, so their errors concentrate on exactly the candidate motions and dense bearings that the planner and the reactive layer rely on, whereas the present-range MLP does not extrapolate along motions it cannot represent. Under this reading, the benefit of the token representation comes from the combination of tokens with query conditioning rather than from the tokens themselves; the unconditioned variant does not improve on the baseline in any column of Table 1.

5.5 Ablation

Table 2 removes one component at a time from the full model.

Removing motion-plan conditioning raises the error to 0.93 m and lowers success by 6.2 percentage points to 85.6%, which is close to the present-range MLP in Table 1 (85.8%). Without the plan, the predictor cannot anticipate the future pose, and this result is consistent with its intended reduction to present-range behavior. Replacing the token decoder with a direct per-ray head attached to the generator raises the error to 0.96 m and lowers success by 3.5 percentage points to 88.3%. This variant retains plan conditioning, and its success remains above the FutureRay-style baseline (79.8%) despite a higher error, which again indicates that the aggregate error does not fully capture the properties that matter for closed-loop avoidance. Reducing the number of tokens from 16 to 4 has the largest effect: the error rises to 1.03 m and success falls by 13.7 percentage points to 78.1%, below every predictor in Table 1. With four tokens, each sector spans a large part of the bearing circle, and the token bottleneck appears to discard angular detail that the decoder cannot recover from the query embedding. Only these two token counts were evaluated, so

the location of the optimum between and beyond them is not known.

5.6 Failure cases

The recorded failure cases identify three distinct mechanisms.

Occluded U-turn (scene 14). In scene 14, the robot must perform a U-turn around an occluding obstacle. Both future predictors, the FutureRay-style baseline and the token predictor, degrade to a range prediction error of 1.6 m in this scene, because the region the robot turns into is not visible in the range history and neither model can infer free space it has not observed. In this situation the reactive layer falls back to present range, and the success rate of the proposed system holds at 84%. The fallback therefore bounds the damage of an unreliable prediction, but it does so by discarding the anticipation that the future predictor is meant to provide.

Planner stall. When the motion-plan input is dropped during execution to simulate a planner stall, the token predictor behaves like the present-range MLP, as intended by plan dropout, and avoidance success decreases by 7.3%. This behavior is a graceful degradation rather than a catastrophic failure, but it shows that a substantial part of the benefit depends on a timely plan. The stall test and the ablation without plan conditioning differ: in the ablation the model is trained without the plan, whereas in the stall test the plan disappears at run time from a model trained to use it.

Thermal throttling. Under thermal throttling on the Orin, the query latency spiked once to 41 ms, which exceeds two control periods of the 50 Hz consumer. The reactive layer re-sampled the most recent prediction at the current time offset and logged the drop, as described in Section 4. The event shows that the 95th-percentile latency does not bound the worst case and that compute thermal state is a relevant operating condition.

06 Limitations

The closed-loop results in this paper are obtained in simulation. Although the latency is measured on the onboard Orin computer, no physical closed-loop obstacle-avoidance trials on ANYmal are reported here, and the range of sensor noise, terrain, and dynamic obstacles encountered on hardware may change both the error and the success rates. The evaluation reports aggregate numbers without per-scene breakdowns or confidence intervals, and differences in prediction error between methods are small (at most 0.08 m among the four predictors in Table 1); the closed-loop success differences are larger, but their scene-level variability is not characterized. The FutureRay-style baseline is a re-implementation within this stack and may not reflect the performance of the original system. The decoder treats plan deviations as small perturbations of the nominal plan used for token generation, which

limits how far the planner can explore without regenerating tokens. The occluded U-turn shows that no amount of query flexibility compensates for unobserved space, and the planner-stall test shows that the advantage of the method depends on a timely plan. Finally, only two token counts were tested, and the latency spike under thermal throttling indicates that worst-case timing guarantees require additional engineering beyond the predictor itself.

07 Conclusion

This paper represented the control-aligned future range field of a quadruped as a small set of latent motion tokens generated from range history and the motion plan, and decoded ray queries on demand from these tokens. The query interface lets a 2 Hz planner and a 50 Hz reactive layer share one predictor at different query densities. In Isaac Lab experiments with ANYmal, the method reached 0.87 m range prediction error, 13.6 ms p95 query latency on the onboard Orin, and 91.8% avoidance success with 1.2 M parameters, against 0.91 m, 14.4 ms, and 79.8% for a FutureRay-style direct-regression baseline. The ablations indicate that the gains depend on plan conditioning, on query-conditioned decoding, and on a sufficient number of tokens. Future work should validate the system in physical closed-loop trials, characterize per-scene variability, extend the token budget study, and address occluded regions, for example by combining the predictor with a memory of previously observed space.

References

- [1] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning Agile and Dynamic Motor Skills for Legged Robots. arXiv:1901.08652, 2019.
- [2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning Quadrupedal Locomotion over Challenging Terrain. arXiv:2010.11251, 2020.
- [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] 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] T. Zang, S. Wang, H. Huang, and S. Cheng. FutureRay: Control-Aligned Future Range for Agile Quadruped Navigation. arXiv:2608.01197, 2026.
- [6] V. Makoviychuk et al. Isaac Gym: High Performance GPU-Based Physics Simulation for Robot Learning. arXiv:2108.10470, 2021.
- [7] M. Mittal et al. Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments. arXiv:2301.04195, 2023.
- [8] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid Motor Adaptation for Legged Robots. arXiv:2107.04034, 2021.
- [9] 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.
- [10] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged Locomotion in Challenging Terrains using Egocentric Vision. arXiv:2211.07638, 2022.
- [11] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme Parkour with Legged Robots. arXiv:2309.14341, 2023.
- [12] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot Parkour Learning. arXiv:2309.05665, 2023.
- [13] D. Hoeller, N. Rudin, D. Sako, and M. Hutter. ANYmal Parkour: Learning Agile Navigation for Quadrupedal Robots. arXiv:2306.14874, 2023.
- [14] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter. Advanced Skills by Learning Locomotion and Local Navigation End-to-End. arXiv:2209.12827, 2022.
- [15] D. Hoeller, L. Wellhausen, F. Farshidian, and M. Hutter. Learning a State Representation and Navigation in Cluttered and Dynamic Environments. arXiv:2103.04351, 2021.
- [16] 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.
- [17] Y. Kim, C. Kim, and J. Hwangbo. Learning Forward Dynamics Model and Informed Trajectory Sampler for Safe Quadruped Navigation. arXiv:2204.08647, 2022.
- [18] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter. Fast Traversability Estimation for Wild Visual Navigation. arXiv:2305.08510, 2023.
- [19] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter. Perceptive Locomotion through Nonlinear Model Predictive Control. arXiv:2208.08373, 2022.
- [20] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena. From Perception to Decision: A Data-driven Approach to End-to-end Motion Planning for Autonomous Ground Robots. arXiv:1609.07910, 2016.
- [21] L. Tai, G. Paolo, and M. Liu. Virtual-to-real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation. arXiv:1703.00420, 2017.
- [22] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis. Learning Navigation Behaviors End-to-End with AutoRL. arXiv:1809.10124, 2018.
- [23] 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.
- [24] G. Kahn, P. Abbeel, and S. Levine. BADGR: An Autonomous Self-Supervised Learning-Based Navigation System. arXiv:2002.05700, 2020.
- [25] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. GNM: A General Navigation Model to Drive Any Robot. arXiv:2210.03370, 2022.
- [26] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine. ViNT: A Foundation Model for Visual Navigation. arXiv:2306.14846, 2023.
- [27] A. Sridhar, D. Shah, C. Glossop, and S. Levine. NoMaD: Goal Masked Diffusion Policies for Navigation and Exploration. arXiv:2310.07896, 2023.
- [28] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. Learning High-Speed Flight in the Wild. arXiv:2110.05113, 2021.
- [29] N. Mohajerin and M. Rohani. Multi-Step Prediction of Occupancy Grid Maps with Recurrent Neural Networks. arXiv:1812.09395, 2018.
- [30] M. Itkina, K. Driggs-Campbell, and M. J. Kochenderfer. Dynamic Environment Prediction in Urban Scenes using Recurrent Representation Learning. arXiv:1904.12374, 2019.
- [31] X. Weng, J. Wang, S. Levine, K. Kitani, and N. Rhinehart. Inverting the Pose Forecasting Pipeline with SPF2: Sequential Pointcloud Forecasting for Sequential Pose Forecasting. arXiv:2003.08376, 2020.

- [32] B. Mersch, X. Chen, J. Behley, and C. Stachniss. Self-supervised Point Cloud Prediction Using 3D Spatio-temporal Convolutional Networks. arXiv:2110.04076, 2021.
- [33] M. Bansal, A. Krizhevsky, and A. Ogale. ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst. arXiv:1812.03079, 2018.
- [34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. arXiv:1706.03762, 2017.
- [35] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh. Set Transformer: A Framework for Attention-based Permutation-Invariant Neural Networks. arXiv:1810.00825, 2018.
- [36] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira. Perceiver: General Perception with Iterative Attention. arXiv:2103.03206, 2021.
- [37] A. Jaegle et al. Perceiver IO: A General Architecture for Structured Inputs and Outputs. arXiv:2107.14795, 2021.
- [38] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-End Object Detection with Transformers. arXiv:2005.12872, 2020.
- [39] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy Networks: Learning 3D Reconstruction in Function Space. arXiv:1812.03828, 2018.
- [40] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. arXiv:1901.05103, 2019.
- [41] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. arXiv:2003.08934, 2020.
- [42] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. arXiv:2006.10739, 2020.
- [43] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering Diverse Domains through World Models. arXiv:2301.04104, 2023.
- [44] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control Barrier Functions: Theory and Applications. arXiv:1903.11199, 2019.
- [45] A. Kendall and Y. Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? arXiv:1703.04977, 2017.