Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Gait-Aware Command Timing for Decentralized Quadruped Swarms

Bishwaswarup Nayak

bishwaswarup@iisc.ac.in

Indian Institute of Science Bangalore https://orcid.org/0009-0001-9926-5329

Research Article

Keywords: Legged robots, Multi-robot systems, Collision avoidance, Gait phase, Quadruped swarms, ORCA

Posted Date: October 13th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11332226/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Gait-Aware Command Timing for Decentralized Quadruped Swarms

Bishwaswarup Nayak [1*]

[1*]Department of Physics, Indian Institute of Science, Bengaluru, 560012, Karnataka, India.

Corresponding author(s). E-mail(s): bishwaswarup@iisc.ac.in;

Abstract

Multi-robot collision avoidance treats each robot as a point that can change velocity at any instant. A trotting quadruped cannot: how well it absorbs a change of commanded velocity depends on where in the gait cycle the change arrives. We measure this dependence for a Bézier trot controller on a simulated Unitree Go1 and build a decentralized swarm layer around it. Each robot solves reciprocal collision avoidance (ORCA) one gait period ahead, holds each non-urgent velocity change for at most half a gait period, releases it at the phase a capability map predicts is least destabilizing, and shapes it with a gait-aware slew limiter; urgent conflicts bypass the hold. In MuJoCo, in crowded or obstructed scenarios with 6, 9, 16 and 32 robots, the full system reduced urgent avoidance events by 22–39% and 99th-percentile body tilt by 19–24% relative to a reactive baseline (paired by seed, Holm-corrected). It had no falls in 230 runs, against two in 170 baseline runs, and fewer robot contacts, though not none. A control that releases after a random wait of the same length shows that prediction and the bounded hold produce the reduction in urgent events, while the phase choice adds a further 4–6% reduction in tilt. Replaying 488 recorded swarm events at nine release times shows that the map's phase ranking, accurate for isolated steps, transfers only partly to the rapid command sequences of a swarm. With four robots the method increases urgent events; with eight it does not significantly change them.

Keywords: Legged robots, Multi-robot systems, Collision avoidance, Gait phase, Quadruped swarms, ORCA

1 Introduction

Decentralized collision avoidance methods such as reciprocal velocity obstacles and ORCA [1, 2] give each robot a new velocity every control tick. They assume that the platform can follow that velocity, possibly subject to an acceleration limit. Wheeled robots and multirotors come close to that assumption. A trotting quadruped does not: its body is supported by two diagonal feet at a time, the support line rotates during each stance, and a velocity change that arrives at the wrong moment produces a large body tilt or a stumble.

That a legged gait responds to a disturbance differently at different phases is well known in single-robot locomotion: push recovery, capture-point and step-timing methods all exploit it [3–6]. What has not been studied, to our knowledge, is the consequence for multi-robot navigation. A swarm layer that issues frequent avoidance commands is a source of disturbances to every robot's gait. Formation and swarm controllers for quadrupeds [7, 8] shape *what* each robot is commanded, but not *when* within the gait the command is delivered.

This paper asks whether timing the swarm layer's velocity changes to the gait phase makes a quadruped swarm safer, and at what cost, and separates two mechanisms that such timing combines: holding a non-urgent change for a short, bounded time, and choosing the gait phase at which it is released. We answer it in simulation with a fully open stack and paired statistics. Our contributions are:

1. A measurement of phase-dependent fragility for a Bézier trot on a Unitree Go1 model: a pilot experiment, a capability map over phase, direction and size of a command change (held-out R^2 0.94–0.98 for isolated steps, against 0.70–0.87 without phase), and a push-impulse envelope in which knowing the phase raises the tolerable impulse by a factor of 1.33–1.44.
2. A decentralized command layer for quadruped swarms: ORCA solved one gait period ahead, a commander that holds each non-urgent change for at most half a gait period and releases it at the phase the map predicts is safest, a gait-aware slew limiter, and an urgent bypass.
3. An ablation and scaling study over 4–32 robots, a gate, pillar fields and a 53 m obstacle course (1,110 runs, paired by seed, Holm-corrected), with controls that release after a random wait of the same length or at the map's worst phase. Prediction and the bounded hold produce the reduction in urgent events; the phase choice adds a smaller, consistent reduction in body tilt.
4. A test of the capability map in the context where it is used: 488 recorded swarm events replayed at nine release times. The map's phase ranking transfers only partly from isolated steps to the rapid command sequences of a swarm, and we identify where it holds.

2 Related work

2.1 Reciprocal collision avoidance

Velocity obstacles and their reciprocal forms [1, 2] are the standard decentralized method for crowds of robots; ORCA reduces each control tick to a small linear

Table 1 How this work relates to the three closest research areas.

Area	What it addresses	What this paper adds
Reciprocal collision avoidance [2, 9, 10]	Decentralized velocity selection for collision avoidance under constant platform limits	Plans one gait period ahead and holds non-urgent changes, releasing them by gait phase
Gait robustness and step timing [3–5]	Phase-dependent balance and recovery of one robot	Uses the measured phase-dependent response to schedule navigation commands
Quadruped formation and navigation [7, 8]	Formation keeping and collision avoidance with velocity-controlled robots	Tests whether command timing improves body stability during team navigation

program. Extensions add platform limits: acceleration-velocity obstacles bound the acceleration [9], and NH-ORCA bounds the tracking error of non-holonomic robots [10]. All of these model the platform's limits as constant in time. We keep ORCA unchanged and instead model a limit that varies periodically with the gait.

2.2 Multi-quadruped formation and navigation

Recent work on quadruped teams uses distributed model predictive control with control barrier functions [7] or leader–follower formations with path planning [8]. These treat each quadruped as a velocity-controlled point and rely on the locomotion controller to follow the commands. The phase-aware layer proposed here is complementary: it can sit between any such planner and the locomotion controller.

2.3 Phase-dependent response of legged gaits

The capture point [3], step-timing adaptation [4] and switched-system capturability for quadrupeds [5] all show that what a legged robot can recover from depends on its state within the step, and gait-sensitivity analysis [6] measures how disturbances propagate through a limit cycle. These results are used to choose footholds and step times for one robot. We use the same fact one level up: to choose *when* the navigation layer changes its command.

2.4 Command shaping for legged robots

Velocity commands to legged controllers are commonly rate-limited, and learned controllers are trained on curricula of command changes. A recent example schedules commands by the robot's current velocity to extend its top speed [11]; the aim there is the locomotion envelope of one robot. Our question differs in purpose: we choose *when* a navigation-layer velocity change, issued for collision avoidance in a team, reaches the locomotion controller. To the best of our knowledge, existing multi-robot navigation approaches do not explicitly schedule navigation-layer command release according to the gait phase of the receiving legged robot..

3 The low-level controller and its phase-dependent fragility

3.1 Controller

Each robot runs the same open-source trot controller. The feet follow a 12-point Bézier swing curve [12, 13] in a diagonal trot (period 0.36 s, duty factor 0.5, swing clearance 0.07 m, hip-to-foot height 0.27 m). A PI loop on body velocity ($k_i = 1.5$) removes steady drift, Raibert foot placement [14] (gain 0.1) corrects velocity errors, and joint targets go to the position actuators of the MuJoCo Menagerie Go1 model [15, 16]. A slew limiter bounds the rate of change of the command at $(1.5\,\mathrm{m/s^2}, 1.0\,\mathrm{m/s^2}, 3.0\,\mathrm{rad/s^2})$ for (v_x, v_y, ω_z). The controller tracks commands to within about 0.02 m/s, survives 4 min of random command changes without a fall, and walks at heights from 0.16 m to 0.34 m.

3.2 Stability

Treating one gait period as a stroboscopic Poincaré map, the largest Floquet multiplier is 0.42–0.59 for the full controller and 0.73–0.87 open loop, so the gait is orbitally stable over the commanded speed range. Stability of the cycle does not say how a single command change is absorbed, which depends on when it arrives.

3.3 Why the phase matters

In a trot the body is supported by one diagonal pair at a time: front-right with rear-left, then front-left with rear-right, at roughly $-34°$ and $+34°$ to the heading for the Go1's geometry. Within one stance the effective support axis rotates from about $+25°$ to $+65°$ as the body moves over the feet. A change of commanded velocity that pushes the body across the current support axis is resisted by the stance legs; one that pushes along it tips the body about that axis. The same change therefore produces different tilt at different phases, and by the mirror symmetry of the trot the pattern repeats every half period with the lateral direction flipped.

3.4 Measured fragility

A pilot experiment applied command steps and body pushes at controlled phases. Phase had a large effect: in 27 of 64 push conditions, whether the robot recovered depended on the phase alone. From these data we built a capability map: a lookup table, periodic in phase and symmetrized, that predicts peak tilt and fall risk for a command change of a given direction and size at a given phase (Fig. 1a). On held-out trials it explains 94–98% of the variance, against 70–87% for the same model without phase. A push-impulse envelope found by bisection (Fig. 1b) agrees with the pilot in 98.6% of cases, and knowing the phase raises the impulse the robot tolerates by a factor of 1.33–1.44.

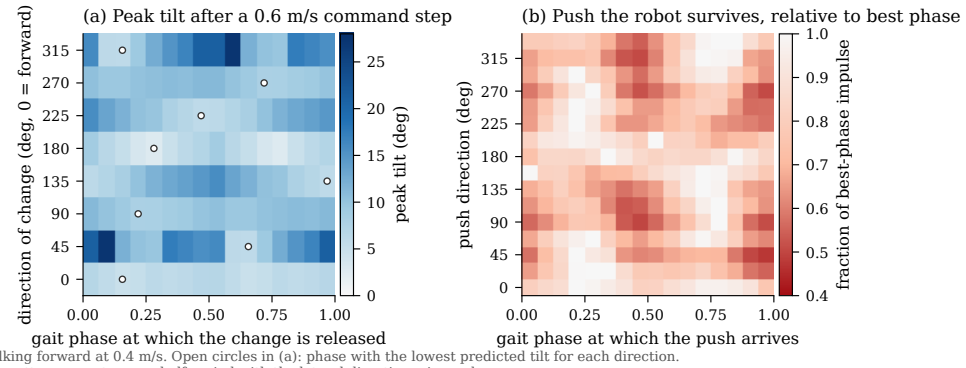


Fig. 1 Phase dependence of the trot controller while walking forward at 0.4 m/s. (a) Peak body tilt after a 0.6 m/s step in commanded velocity, by direction of the change and the gait phase at which it is released; open circles mark the best phase for each direction. (b) Largest push impulse the robot survives, as a fraction of the value at its best phase for that direction.

3.5 What this means for a swarm

In isolated step tests, releasing a command at the best phase strictly dominated the slew limiter alone. In a stream of swarm commands, however, timing alone did not work: only 0–2% of the commands resembled the settled single steps the map was measured on, and about half were urgent, leaving no time to wait for a good phase. Section 4 is designed around that finding: the swarm layer must look ahead so that most changes are known before they are needed.

4 Method

Every robot runs the same four-stage stack at 10 Hz (Fig. 2), with no central planner and no communication beyond the neighbour positions and velocities that ORCA needs. A safety layer that is identical in every configuration, the baseline included, wraps the stack.

4.1 Predictive swarm layer

Each robot is a disc of radius 0.38 m with maximum speed 0.6 m/s. Rather than solving ORCA for the present, the robot solves it for the situation one lookahead L ahead ($L = 0.36$ s, one gait period). Its own position is extrapolated with the command it is currently executing, and its neighbours with their measured velocities. The result is the velocity the robot will need at time $t + L$, so it can be requested with a deadline of L. If the current command would collide within 0.8 s with a robot, or within 1.5 s with a static obstacle, the request is urgent: ORCA is solved for the present and the command is released at once. A 15° keep-right bias breaks symmetric deadlocks. Each robot sees the positions and velocities of up to ten neighbours within 4 m, without noise or delay; velocities are estimated from positions with a 0.15 s low-pass filter.

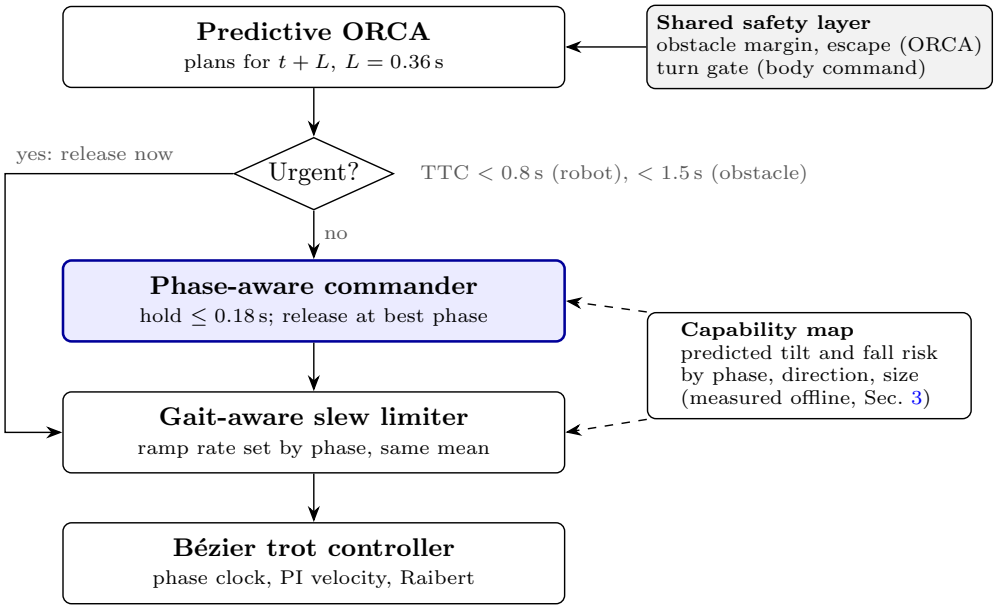


Fig. 2 Per-robot stack. ORCA plans one gait period ahead; non-urgent changes wait in the commander for the phase the capability map predicts is safest, urgent ones go straight to the limiter. The safety layer is the same in every configuration.

4.2 Phase-aware commander

A request whose change from the current command exceeds 0.15 m/s (or 0.3 rad/s) is a manoeuvre. A non-urgent manoeuvre is held for at most half a gait period (0.18 s), which is always shorter than its deadline L. Over that window the commander queries the capability map at 1/64-period steps and releases the command at the phase with the lowest predicted cost, namely peak tilt plus a weight of 50 on the predicted fall probability. Smaller changes, urgent requests and changes outside the range of the map's grid of speeds, directions and sizes are released at once. Every other manoeuvre is timed, including the many that occur during walking unlike the settled, straight steps on which the map was measured (Section 6.3 tests the map on exactly these). A new request similar to a pending one updates the pending command but keeps its release time, so the timing is not reset every tick.

4.3 Gait-aware slew limiter

Once released, a change is applied through a slew limiter whose rate depends on phase. The limiter is built from the capability map: it ramps faster in phases where the map predicts little tilt for the current direction of change and slower where it predicts a lot. Its rate is scaled so that its mean over one period equals that of the uniform limiter, so it does not gain speed by being more aggressive on average.

4.4 Formation and queueing

Formations follow a reference point with slot offsets that shrink laterally when obstacles are close ahead. In bottlenecks, the robot further back takes 90% of the avoidance responsibility in ORCA and the robot in front 10%, so that robots queue instead of shouldering through.

4.5 Shared safety layer

In every configuration, ORCA sees static obstacles inflated by 0.1 m with a 2 s horizon. Body commands are clipped to the range in which the gait was stress-tested ($|\omega_z| \leq 1.0\,\mathrm{rad/s}$, $|v_y| \leq 0.3\,\mathrm{m/s}$), and forward speed is scaled from 100% to 30% as the heading error grows from 30° to 90°, so a robot turns before it walks. If ORCA returns less than 0.05 m/s for more than 1 s while the robot wants to move, the preferred velocity is turned 75° towards the formation reference for 2 s, and to the other side if the robot is still stuck. Each of these was added after a failure found in the experiments; Section 7 reports one further fix that we tried and rejected.

4.6 Scope of the safety layer

ORCA returns velocities that are collision-free for ideal agents that adopt them at once and keep them. A quadruped does neither, and the stack handles the gap heuristically rather than with a guarantee. (i) Future positions are predicted by constant-velocity extrapolation over L: the robot's own position with the command it is executing, its neighbours' with their measured velocities. (ii) The selected velocity becomes the active command after the heading controller and the body-command limits (Section 4), when the commander releases it, at most 0.18 s after the request; the slew limiter then ramps the gait towards it. (iii) The prediction does not model the robot's own tracking error: the lag of the ramp and the controller's tracking error enter only through the neighbours' measured velocities and through replanning every 0.1 s. (iv) A release cannot miss its deadline, because the longest hold (0.18 s) is shorter than L; the ramp after release can, however, extend past $t + L$. (v) A request is urgent, and bypasses both prediction and the hold, when the time to collision of the current command is below 0.8 s with a robot or 1.5 s with a static obstacle. (vi) The safety layer therefore reduces collision risk but does not ensure it: the full system had 16 robot–robot and 1 robot–obstacle contact onsets in 230 runs (Section 6.8). The collision avoidance reported here is validated empirically, not proven.

5 Experimental setup

All experiments run in MuJoCo [15] with N copies of the Menagerie Go1 model [16] in one scene (foot friction 0.9). The swarm layer runs at 10 Hz and the gait controller at the physics rate. Each seed perturbs every start position by 5 cm (standard deviation) and gives every robot a random initial gait phase, so robots are never phase-locked. Table 2 lists the scenarios and Table 3 the configurations.

The main suite runs all six configurations on the 4- and 8-robot swaps, the 9-robot pillar field and the gate with 20 seeds (480 runs). The scaling suite runs the

Table 2 Scenarios.

Scenario	Robots	Layout	Tests
Swap	4 / 8 / 16 / 32	On a circle of radius 2.5 / 4 / 6 / 9 m; each robot walks to the opposite point	Crossing traffic; density grows with N
Pillars	9 (3×3, 1.4 m spacing)	Formation travels 14 m at 0.35 m/s through 5 pillars ($r = 0.3$ m)	Formation through clutter
Gate	6 (2×3)	Wall with a 1.9 m opening	Bottleneck: the team narrows and queues
Pillars	16 (4×4)	18 m through 9 pillars	Dense formation through clutter
Course	1 and 6	53 m path at 0.45 m/s: pole slalom, 1.9 m gate, chicane, two obstacles moving across the path (0.4 m/s), 7 pillars	All of the above in sequence

Table 3 Configurations. All share the safety layer of Section 4. The last two are controls: they hold each change for the same window as the full system but release it after a uniformly random wait, or at the phase the map predicts is worst.

Configuration	Prediction	Timing	Limiter	Queue
Baseline (reactive)	–	–	–	–
Queueing only	–	–	–	✓
Prediction only	✓	–	–	–
Prediction + timing	✓	✓	–	–
+ gait-aware limiter	✓	✓	✓	–
Full system	✓	✓	✓	✓
Full system, random wait (sham)	✓	random	✓	✓
Full system, worst phase	✓	worst	✓	✓

baseline, prediction only (with queueing) and the full system on the 16- and 32-robot swaps and the 16-robot pillar field with 10 seeds (90 runs). The lookahead suite varies $L \in \{0.18, 0.36, 0.54, 0.72\}$ s on the 8-robot swap and the gate (10 seeds, 100 runs). The course runs three configurations for teams of 1 and 6 robots with 20 seeds (120 runs). The timing-control suite runs the baseline, the full system and its two controls on the four main scenarios with 20 seeds (320 runs); the mean hold was 0.08–0.09 s in all three timed configurations.

5.1 Map validation in context

To test the capability map where it is used, we run the full system on the 8- and 16-robot swaps, the 9-robot pillar field and the gate (5 seeds each) and sample one in four of the manoeuvres the commander times, at most 30 per run (488 events). At each sampled request the complete simulation state is saved. From it, the same change is released after each of nine waits from 0 to half a gait period, while every robot's logged command stream continues as recorded, and the peak tilt of the focal robot

over the next 1.0 s is measured. Replaying the commander's own choice reproduces the original run exactly in every event. For each event this gives a measured tilt at every candidate phase, so the map can be scored on absolute error, on how well it ranks the phases, and on the tilt of the phase it would choose.

5.2 Metrics

Three metrics are fixed in advance as primary: urgent avoidance events per robot-minute (how often a robot had to change course with less than 0.8 s to spare), the 99th-percentile body tilt over the run, and the closest robot–robot distance. We also report falls (trunk below 0.15 m or tilted more than 60°), runs not finished within the time limit, contacts and completion time.

5.3 Statistics

Comparisons are paired by seed. For each metric we take the per-seed difference and test it with an exact two-sided sign-flip permutation test that enumerates all 2^n sign patterns (n = number of seeds); zero differences are kept and do not change the result, and ties with the observed statistic count as extreme. We also report 95% bootstrap confidence intervals. Three families of tests are corrected separately with the Holm method. (A) The 21 primary tests compare the full system with the reactive baseline (7 scenarios × 3 metrics). (B) Twelve tests compare prediction with timing against prediction alone on the four main scenarios. (C) The timing controls compare the full system with its random-wait and worst-phase versions. With $n = 10$ seeds the smallest possible raw p-value is $2/2^{10} \approx 0.00195$; Holm multiplies the k-th smallest of the 21 primary p-values by $22 - k$, so a 10-seed test ranked first could at best reach $21 \times 0.00195 \approx 0.041$, below 0.05 but never below 0.01. Holm is a step-down procedure: the adjusted value of the k-th test is the largest of $(22 - j)\, p_{(j)}$ over $j \le k$, capped at 1. In our data nine of the twelve 20-seed tests have smaller p-values than any 10-seed test, so the six significant 10-seed tests, tied at $p = 2/2^{10}$, occupy ranks 10 to 15 and all receive the adjusted value $12 \times 0.00195 \approx 0.023$ of the first of them. Table A1 in the appendix lists all 21 raw and adjusted p-values in order. The scaling suite can therefore reach the 5% level but not the 1% level.

6 Results

We report three comparisons. (A) The full system against the reactive baseline measures the value of the whole architecture (Fig. 3, Table 4). Of its 21 tests, 15 are significant after Holm correction: 13 favour the full system and 2, both in the 4-robot swap, favour the baseline. (B) Prediction with timing against prediction alone isolates the commander (Section 6.1). (C) The full system against two controls that hold each change for the same window but release it after a random wait or at the map's worst phase isolates the phase choice itself (Section 6.2).

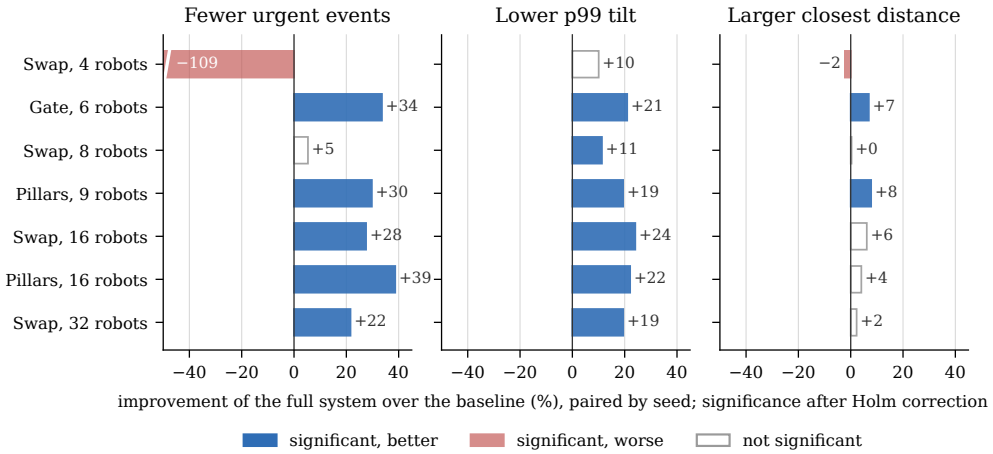


Fig. 3 Improvement of the full system over the baseline on the three primary metrics. Filled bars are significant after Holm correction over all 21 tests. The 4-robot swap bar for urgent events is cut at −50% (true value −109%).

Table 4 Primary metrics, baseline (B) against full system (F), means over seeds. $^{*}\,p < 0.05$, $^{**}\,p < 0.01$ after Holm correction over all 21 tests.

			Urgent events / robot-min			p99 tilt (deg)			Closest distance (m)		
Scenario	N	Seeds	B	F	Change	B	F	Change	B	F	Change
Swap	4	20	5.2	10.9	+109%**	5.90	5.30	−10%	0.744	0.727	−2%**
Gate	6	20	24.4	16.2	−34%**	5.84	4.62	−21%**	0.661	0.707	+7%**
Swap	8	20	29.6	28.0	−5%	7.37	6.54	−11%**	0.626	0.628	+0%
Pillars	9	20	40.4	28.4	−30%**	7.14	5.76	−19%**	0.604	0.651	+8%**
Swap	16	10	54.1	39.2	−28%*	8.51	6.46	−24%*	0.600	0.637	+6%
Pillars	16	10	42.0	25.7	−39%*	7.22	5.62	−22%*	0.593	0.617	+4%
Swap	32	10	64.5	50.6	−22%*	8.39	6.76	−19%*	0.566	0.578	+2%

6.1 Which component does the work

Prediction alone does not help (Fig. 4). Against the baseline it changes urgent events by +6% in the 8-robot swap and +3% in the pillar field, and raises 99th-percentile tilt by 20% in the 4-robot swap. Adding the commander, which holds each non-urgent change and releases it at the predicted best phase, produces most of the gain (Table 5): against prediction alone it lowers urgent events by 18–20% in the 8-robot swap and the pillar field and 99th-percentile tilt by 12–23% in all four scenarios. The gait-aware limiter adds further tilt reduction at the gate (−15% instead of −12% against the baseline) but almost none elsewhere. Queueing matters in obstructed formations, where it doubles the reduction in urgent events (pillars −30% instead of −18%, gate −34% instead of −14%) and widens the closest distance by 7–8%. The commander combines a bounded hold with a choice of phase; Section 6.2 separates the two.

Table 5 Comparison B: prediction with timing (P+T) against prediction alone (P), means over 20 seeds. $^{*}p < 0.05$, $^{**}p < 0.01$ after Holm correction over these 12 tests. Closest distance changed by +1 to +5%, none significant.

	Urgent events / robot-min			p99 tilt (deg)		
Scenario	P	P+T	Change	P	P+T	Change
Swap, 4	13.6	10.8	−21%	7.09	5.45	−23%**
Swap, 8	31.4	25.2	−20%**	7.54	6.52	−14%*
Pillars, 9	41.5	34.0	−18%**	6.97	6.03	−13%**
Gate, 6	21.6	20.9	−3%	5.82	5.14	−12%**

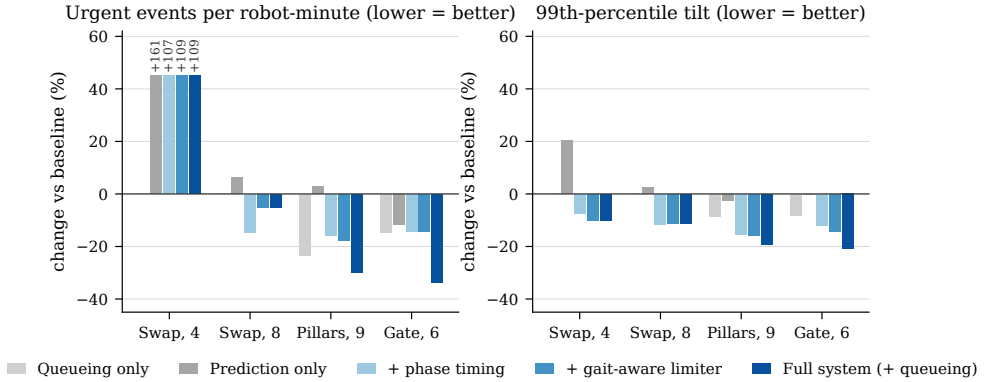


Fig. 4 Ablation on the main suite (20 seeds). Change relative to the reactive baseline as components are added; lower is better. Prediction alone does not help; adding the commander (hold and phase choice) produces most of the gain.

6.2 Phase choice or waiting

The commander's benefit could come from releasing at the best phase or merely from holding the change for a fraction of a gait period. The two controls hold for the same mean time (0.08–0.09 s) and differ only in when they release (Fig. 5, Table 6). Urgent events and closest distance do not depend on the release phase: none of the 16 tests on these metrics is significant, and the random-wait control alone reduces urgent events against the baseline by 27% in the pillar field and 30% at the gate. The reduction in urgent events therefore comes from prediction and the bounded hold, not from the phase. Tilt does depend on it, modestly and with the expected ordering: in every scenario the map's best phase gives the lowest 99th-percentile tilt and its worst phase the highest. After Holm correction the best phase beats the random wait in the 4-robot swap (−11%) and the worst phase in the 4-robot swap and the pillar field (−21%, −8%). Pooled over the 80 seed pairs (an exploratory analysis), the best phase lowers p99 tilt by 5.9% against the random wait ($p < 10^{-4}$) and the random wait by 4.4% against the worst phase ($p = 0.0005$). In the crowded scenarios the phase choice accounts for 11–28% of the full system's reduction in p99 tilt.

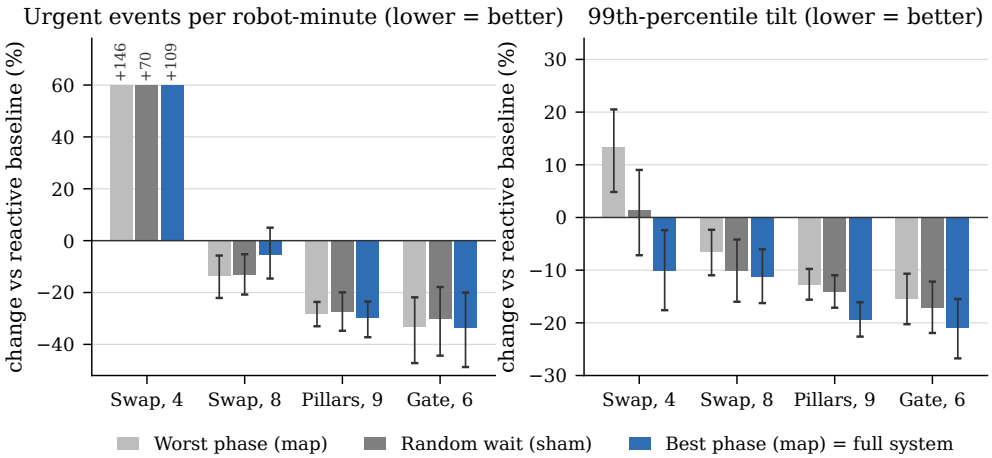


Fig. 5 Timing controls (20 seeds). All three configurations are the full system with the same hold; they differ only in when the held change is released. Bars: mean change against the reactive baseline, 95% bootstrap intervals. Off-scale bars (4-robot swap) are labelled with their value.

Table 6 Comparison C: the full system (map's best phase) against a random wait from the same window and against the map's worst phase, change in the full system, 20 seeds. $^{*}\,p < 0.05$, $^{**}\,p < 0.01$ after Holm correction over 24 tests (closest distance included; all its changes are within ±4% and not significant).

	vs random wait		vs worst phase	
Scenario	Urgent	p99 tilt	Urgent	p99 tilt
Swap, 4	+23%	−11%*	−15%	−21%**
Swap, 8	+9%	−1%	+10%	−5%
Pillars, 9	−3%	−6%	−2%	−8%**
Gate, 6	−5%	−5%	−1%	−7%

6.3 The capability map inside the swarm

In the replay experiment (Section 5.1), 2% of the 488 events met the conditions under which the map was measured, and 92% followed the previous command change by less than one gait period. In this context the map's absolute predictions do not hold (Fig. 6a): it under-predicts peak tilt by 1.8° on average (mean absolute error 2.1°, $R^2 = -0.99$, no better than the phase-blind map), against $R^2 = 0.98$ for isolated steps. Its ranking of phases survives in part. The median rank correlation between predicted cost and measured tilt over the nine release times is +0.27 (positive in 61% of events), and the phase the map chooses has 0.21° less tilt than a random release (95% CI 0.15–0.27°) and 0.37° less than its worst phase, which is 35% of the reduction an oracle choosing the best of the nine phases would achieve. The value of the map's choice falls with distance from the calibration conditions (Fig. 6b): it is 0.95° when the previous change was at least one period ago and 0.97° for the largest changes (map

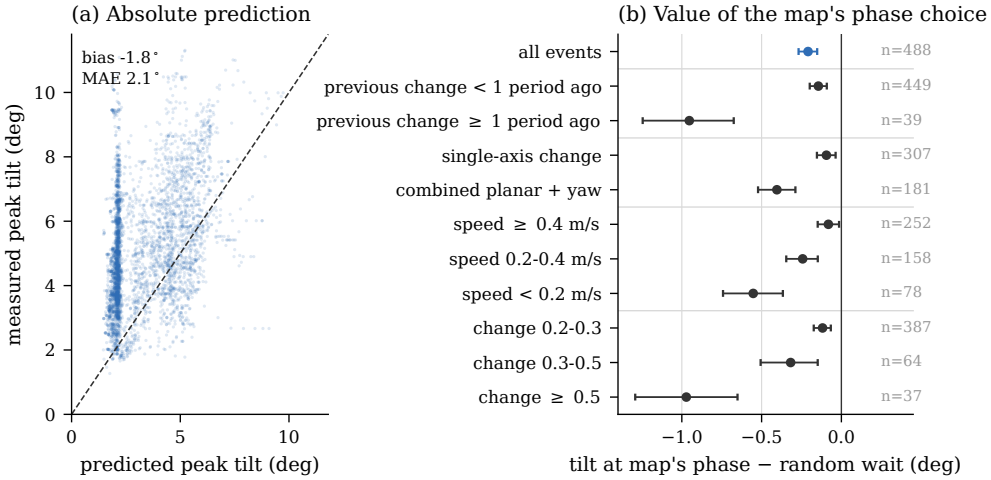


Fig. 6 The capability map in the context where it is used: 488 swarm events, each replayed at nine release times. (a) Predicted against measured peak tilt over 1 s; dashed line: perfect prediction. (b) Tilt at the phase the map chooses minus the mean over the nine release times (a random release), by context; mean and 95% bootstrap interval. The map's choice helps most where events resemble its calibration.

size ≥ 0.5), but 0.14° during rapid command sequences and 0.08° above 0.4 m/s. No replayed release led to a fall or to a tilt above 15°, so fall-risk prediction could not be tested in context.

6.4 Scaling

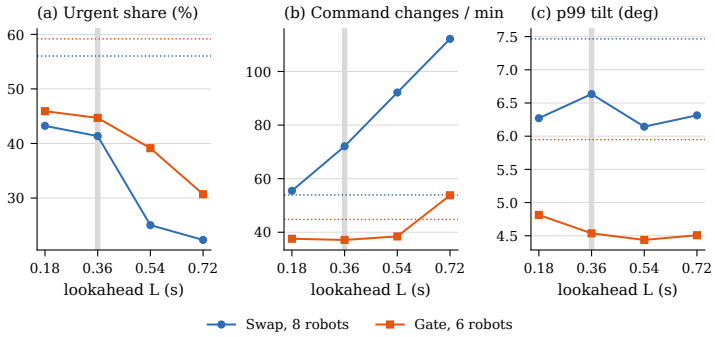
The benefit holds as the team grows. With 16 robots in a swap the full system cuts urgent events by 28% and tilt by 24%; with 32 robots, by 22% and 19%; with 16 robots in a formation through pillars, by 39% and 22%. The gain in closest distance shrinks at 32 robots (+2%, not significant), where the swarm is near its packing limit.

6.5 Cost

Planning ahead and holding commands cost time and, in sparse traffic, safety margin. In the 4-robot swap the full system is 6.8% slower, has twice the urgent events of the baseline (5.2 against 10.9 per robot-minute) and a 2% smaller closest distance. In the 8-robot swap it is 2.7% slower; at the gate the completion time is unchanged. The random-wait control has the same cost (completion times within 0.6% of the full system, not significant), so the cost belongs to the hold, not to the phase choice.

6.6 Lookahead

Looking further ahead lets more changes be planned (Fig. 7): the share of manoeuvres that are urgent falls from 56% (baseline) to 43%, 41%, 25% and 22% for $L = 0.18$, 0.36, 0.54 and 0.72 s in the 8-robot swap. But the number of command changes per robot-minute doubles from 54 to 112, because predictions made further ahead are



Solid: full system at each lookahead; dotted: reactive baseline of the same colour. Grey band: L = 0.36 s (one gait period), used in all other experiments.

Fig. 7 Effect of the lookahead L (10 seeds). Longer lookahead makes fewer manoeuvres urgent (a) but increases command churn (b); tilt is reduced about equally at every L (c).

revised more often. Tilt is reduced about equally at every L. One gait period (0.36 s) is the setting we use; 0.54 s is slightly better at the gate but costs much more command churn in swaps.

6.7 Obstacle course

For the team of six, the full system lowered 99th-percentile tilt from 7.95° to 6.68° (−16%), the worst tilt over 20 runs from 18.3° to 13.7°, urgent events by 14% and robot–robot contacts from 0.25 to 0.05 per run, at the same completion time (129 s). Fig. 8 shows one seed. One of its 20 runs did not finish in time; the baseline had one fall. For a single robot the three configurations are nearly identical (tilt 4.7–4.9°), and prediction raised urgent events (7.4 to 10.4 per minute), consistent with the 4-robot swap.

6.8 Safety totals

Across all 230 full-system runs in the final configuration there were no falls; the baseline fell twice in 170 runs (once in the 16-robot pillar field and once on the course). The full system had 16 robot–robot and 1 robot–obstacle contact onsets in those runs, against 28 and 4 for the baseline: contacts are rarer but not eliminated (Section 4.6). Runs that did not finish in time were rare in every configuration: one for the full system (course), one for queueing only, two for prediction only and one for the worst-phase control.

7 Discussion and limitations

7.1 When it helps

Planning ahead and holding commands pay off when a robot has many manoeuvres to make and enough warning to plan them. Crowds and clutter provide both; sparse traffic does not. In the 4-robot swap the few manoeuvres are mostly avoidable by acting slightly earlier, and any hold only delays them, which shows up as more urgent

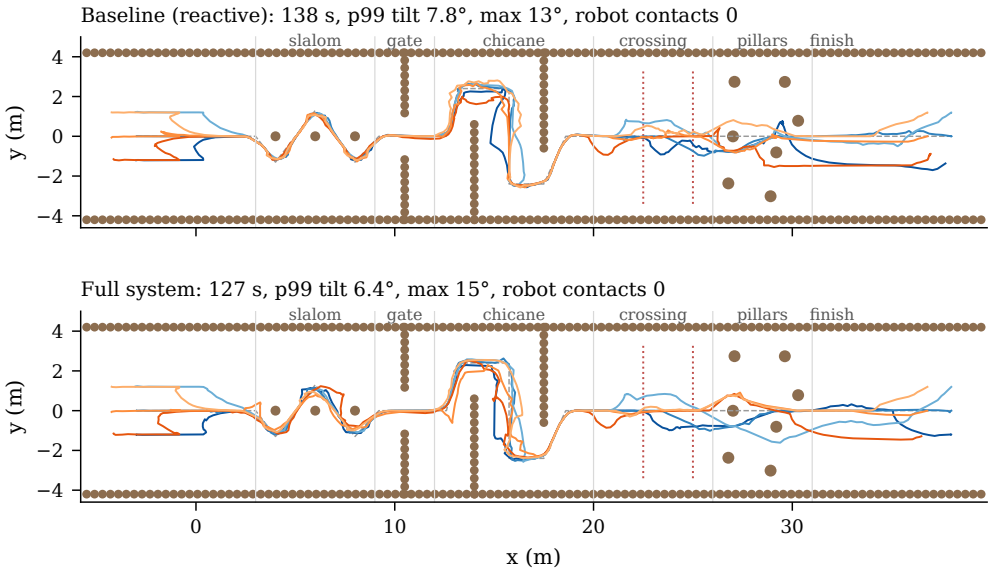


Fig. 8 Paths of a six-robot team on the obstacle course (seed 0) for the baseline (top) and the full system (bottom). Dotted red lines: tracks of the two moving obstacles; dashed grey line: reference path.

events later; the random-wait control shows the same cost, so it belongs to the hold rather than to the phase choice. A practical system should enable the hold only above a traffic-density threshold; we have not tested such a switch.

7.2 What the phase adds

The controls separate two effects that the commander combines. Holding a non-urgent change for a fraction of a gait period, with ORCA planning one period ahead, produces the reduction in urgent events and most of the reduction in tilt; we have not isolated why the hold helps. Releasing at the phase the map predicts is safest adds a smaller, consistent reduction in tilt (about 4–6%), and releasing at the worst phase costs about as much, so the phase matters to body stability but not to how often robots must swerve. The replay explains why the phase effect is small: the map was measured on isolated steps from settled walking, and its ranking holds well only for events that resemble those steps, which are a minority in a swarm. A map measured on command sequences, with the time since the previous change as an additional input, is the natural next step.

7.3 Robustness fixes and a rejected one

The shared safety layer was not designed in advance. Each part was added after a failure found by the experiments: robots falling after combined sideways walking and fast turning (turn gate), robots pinned against a wall between them and their slot (escape), and robots grazing pillars (static margin). Because the layer is identical in every configuration, it does not favour the full system. We also tried a stronger escape

rule, which detects a stall by lack of progress instead of low speed and gives late robots right of way. It removed every fall, but runs that did not finish in time rose from 1 to 16 of 480 in the main suite, because the rule also fired during normal avoidance and sent robots on long detours. We report it as a negative result; the code keeps it as an option.

7.4 Limitations

(i) All results are in simulation, with one robot model and one trot controller; the capability map must be re-measured for another controller, and the effect size on hardware is unknown. (ii) The capability map is measured on single command changes in settled walking; inside the swarm its absolute predictions do not hold and its phase ranking holds only in part (Section 6.3). (iii) Collision avoidance is validated empirically, not guaranteed: the prediction ignores the robot's own tracking error and contacts still occur (Section 4.6). (iv) Robots share positions and velocities perfectly, without noise or delay. (v) The scaling suite has only 10 seeds, which limits the attainable significance, and the pooled analysis of the timing controls is exploratory. (vi) Completion time rises by up to 7% in sparse scenarios. (vii) Falls were too rare to test the map's fall-risk predictions in context.

8 Conclusion

A quadruped's tolerance of a velocity change depends on where in the gait cycle the change arrives. A swarm layer that plans ORCA one gait period ahead, holds each non-urgent change for at most half a period and releases it at the predicted best phase cut urgent avoidance events by 22–39% and 99th-percentile tilt by 19–24% against a reactive baseline in crowded and obstructed scenarios with 6, 9, 16 and 32 simulated Go1 robots, with no falls in 230 runs against two in 170 for the baseline. Controls show that planning ahead and the bounded hold account for the reduction in urgent events, and that the phase choice adds a smaller, consistent reduction in tilt. A capability map measured on isolated steps keeps only part of its phase ranking in the rapid command sequences of a swarm. Next steps are a map measured on command sequences, hardware experiments on Go1 robots, a second low-level controller (a learned policy), sensing noise and delay, and an automatic switch that disables the hold in sparse traffic.

Supplementary information. Online Resource 1 (MP4) shows, side by side, the reactive baseline and the full system in the 16-robot swap, the nine-robot formation through pillars, the six-robot formation through the gate and the six-robot team on the obstacle course (seed 0).

Acknowledgements. None.

Declarations

Funding. The author did not receive support from any organization for the submitted work.

Competing interests. The author has no relevant financial or non-financial interests to disclose.

Use of AI-assisted tools. The author used Claude as an assistant for editing manuscript text, for auditing the numerical claims of the manuscript against the code output. The author conceived and directed the study, chose the models and parameters, ran all computations and interpreted the results, and reviewed and verified all AI-assisted text, including through the automated check of every number in the paper described below. The author takes full responsibility for the content of this manuscript.

Data and code availability. The simulation code, configuration files, raw results and the scripts that produce every table and figure are publicly available at https://github.com/Bishwaswarup/robot (MIT licence). Tables 4, 5 and 6 are generated by `scripts/paper_tables.py` and the figures by `scripts/paper_figures.py` from the raw results in the repository.

Author contributions. B.N. conceived the study, developed the software, ran the experiments, analysed the results and wrote the manuscript.

Ethics approval and consent to participate. Not applicable.

Appendix A Holm correction of the primary tests

Table A1 All 21 primary tests (full system against the reactive baseline) in order of raw p-value, with the Holm multiplier $22 - k$ and the adjusted p-value (the running maximum of $(22 - j)\, p_{(j)}$, $j \le k$). Ranks 1–15 are significant at 0.05; the changes at ranks 7 and 8 (4-robot swap) favour the baseline. Ties are ordered arbitrarily; tied tests receive the same adjusted value.

k	Scenario	N	Seeds	Metric	Change	Raw p	$22 - k$	Holm p
1	Pillars	9	20	Urgent events	−30%	1.9×10^{-6}	21	4.0×10^{-5}
2	Pillars	9	20	p99 tilt	−19%	1.9×10^{-6}	20	4.0×10^{-5}
3	Gate	6	20	Closest distance	+7%	3.8×10^{-6}	19	7.2×10^{-5}
4	Gate	6	20	p99 tilt	−21%	5.7×10^{-6}	18	1.0×10^{-4}
5	Gate	6	20	Urgent events	−34%	7.1×10^{-5}	17	0.00120
6	Pillars	9	20	Closest distance	+8%	3.4×10^{-4}	16	0.00543
7	Swap	4	20	Closest distance	−2%	4.0×10^{-4}	15	0.00601
8	Swap	4	20	Urgent events	+109%	5.0×10^{-4}	14	0.00702
9	Swap	8	20	p99 tilt	−11%	7.7×10^{-4}	13	0.00999
10	Swap	16	10	Urgent events	−28%	0.00195	12	0.0234
11	Swap	16	10	p99 tilt	−24%	0.00195	11	0.0234
12	Swap	32	10	Urgent events	−22%	0.00195	10	0.0234
13	Swap	32	10	p99 tilt	−19%	0.00195	9	0.0234
14	Pillars	16	10	Urgent events	−39%	0.00195	8	0.0234
15	Pillars	16	10	p99 tilt	−22%	0.00195	7	0.0234
16	Swap	16	10	Closest distance	+6%	0.0176	6	0.105
17	Swap	4	20	p99 tilt	−10%	0.0200	5	0.105
18	Pillars	16	10	Closest distance	+4%	0.0566	4	0.227
19	Swap	32	10	Closest distance	+2%	0.293	3	0.879
20	Swap	8	20	Urgent events	−5%	0.306	2	0.879
21	Swap	8	20	Closest distance	+0%	0.895	1	0.895

References

[1] Berg, J., Lin, M.C., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In: 2008 IEEE International Conference on Robotics and Automation, pp. 1928–1935 (2008). https://doi.org/10.1109/ROBOT.2008.4543489

[2] Berg, J., Guy, S.J., Lin, M.C., Manocha, D.: Reciprocal n-body collision avoidance. In: Robotics Research: The 14th International Symposium ISRR 2009. Springer Tracts in Advanced Robotics, vol. 70, pp. 3–19. Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19457-3_1

[3] Pratt, J., Carff, J., Drakunov, S., Goswami, A.: Capture point: A step toward humanoid push recovery. In: Proceedings of the 2006 6th IEEE-RAS International Conference on Humanoid Robots, pp. 200–207 (2006). https://doi.org/10.1109/ICHR.2006.321385

[4] Khadiv, M., Herzog, A., Moosavian, S.A.A., Righetti, L.: Step timing adjustment: A step toward generating robust gaits. In: Proc. IEEE-RAS International Conference on Humanoid Robots (2016). arXiv:1610.02377

[5] Chen, H., Hong, Z., Yang, S., Wensing, P.M., Zhang, W.: Quadruped Capturability and Push Recovery via a Switched-Systems Characterization of Dynamic Balance. arXiv:2201.11928 (2022)

[6] Hobbelen, D.G.E., Wisse, M.: A disturbance rejection measure for limit cycle walkers: The gait sensitivity norm. IEEE Transactions on Robotics **23**(6), 1213–1224 (2007) https://doi.org/10.1109/TRO.2007.904908

[7] Zhan, W., Liang, Z., Song, H., Pan, W.: Safe Distributed Learning-Enhanced Predictive Control for Multiple Quadrupedal Robots. arXiv:2503.05836 (2025)

[8] Norén, M., Stamatopoulos, M.-N., Banerjee, A., Nikolakopoulos, G.: Role-Adaptive Collaborative Formation Planning for Team of Quadruped Robots in Cluttered Environments. arXiv:2602.18260 (2026)

[9] Berg, J., Snape, J., Guy, S.J., Manocha, D.: Reciprocal collision avoidance with acceleration-velocity obstacles. In: 2011 IEEE International Conference on Robotics and Automation, pp. 3475–3482 (2011). https://doi.org/10.1109/ICRA.2011.5980408

[10] Alonso-Mora, J., Breitenmoser, A., Rufli, M., Beardsley, P., Siegwart, R.: Optimal reciprocal collision avoidance for multiple non-holonomic robots. In: Distributed Autonomous Robotic Systems: The 10th International Symposium. Springer Tracts in Advanced Robotics, vol. 83, pp. 203–216. Springer, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-32723-0_15

[11] Tao, Y., Cheng, S., Lan, G., Yuan, Y., Jin, Y., Wang, H.: Extending the Speed Limit of Quadrupedal Locomotion via Refined Actuator Modeling and Adaptive Command Scheduling. arXiv:2609.13289 (2026)

[12] Hyun, D.J., Seok, S., Lee, J., Kim, S.: High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the MIT Cheetah. The International Journal of Robotics Research **33**(11), 1417–1445 (2014) https://doi.org/10.1177/0278364914532150

[13] Rahme, M., Abraham, I., Elwin, M.L., Murphey, T.D.: SpotMiniMini: PyBullet Gym Environment for Gait Modulation with Bezier Curves. GitHub repository. MIT License (2020). https://github.com/OpenQuadruped/spot_mini_mini

[14] Raibert, M.H.: Legged Robots That Balance. MIT Press, Cambridge, MA (1986)

[15] Todorov, E., Erez, T., Tassa, Y.: MuJoCo: A physics engine for model-based control. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5026–5033 (2012). https://doi.org/10.1109/IROS.2012.6386109 . IEEE

[16] Zakka, K., Tassa, Y., MuJoCo Menagerie Contributors: MuJoCo Menagerie: A Collection of High-Quality Simulation Models for MuJoCo. GitHub repository, https://github.com/google-deepmind/mujoco_menagerie (2022)

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- course.mp4
- course.mp4
- pillars.mp4
- swap16.mp4