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.

Learning the Right Abstraction: Neural Reduced Dynamics for Complex Robot Control

Harry Zhang

hzhang699@wisc.edu

University of Wisconsin–Madison

Dan Negrut

University of Wisconsin–Madison

Research Article

Keywords: Neural reduced dynamics models, Off-road autonomy, Reinforcement learning, High-fidelity simulation, Learned dynamics models

Posted Date: October 6th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11068526/v1

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

Additional Declarations: No competing interests reported.

Learning the Right Abstraction: Neural Reduced Dynamics for Complex Robot Control

Harry Zhang[1*] and Dan Negrut[1]

[1*]Department of Mechanical Engineering, University of Wisconsin–Madison, Madison, 53706, WI, USA.

*Corresponding author(s). E-mail(s): hzhang699@wisc.edu;
Contributing authors: negrut@wisc.edu;

Abstract

High-fidelity simulators for embodied artificial intelligence evaluate complex robotic systems realistically, but their computational cost rules out the large-scale reinforcement learning campaigns that modern control policies require. We argue that the practical value of a learned dynamics model lies in learning the *right abstraction*: a reduced state that preserves the control-relevant physics of the high-fidelity system while enabling high-throughput policy learning. We develop a neural reduced dynamics (NRD) framework that separates the state the model propagates from what can be supplied as an input or recovered analytically, trains policies entirely inside the frozen learned model, and validates them back in the high-fidelity simulator. Two case studies instantiate the framework across three control tasks: terrain-aware trajectory tracking for a High Mobility Multipurpose Wheeled Vehicle (HMMWV) on rigid, bumpy, and deformable Continuum Representation Model (CRM) terrain; and goal reaching for a stock tracked vehicle and its front-mounted articulated arm. Every policy transfers back to the high-fidelity simulator. A single policy trained inside the terrain-conditioned model, and given no terrain input of its own, attains lower median and mean tracking error than both single-terrain specialists on all three terrains, including zero-shot bumpy terrain. The tracked vehicle reaches **100** of **100** goals and the arm **97** of **100**, with no contacts or joint-limit violations. The NRD models advance roughly four orders of magnitude faster in simulated time than the high-fidelity scenes they replace, which makes iterative on-policy learning practical. Code, data, and the policy-training scripts are available at https://uwsbel.github.io/NeDM/.

Keywords: Neural reduced dynamics models, Off-road autonomy, Reinforcement learning, High-fidelity simulation, Learned dynamics models

Article Highlights

- Costly physics simulations are distilled into fast learned models that keep the physics a controller actually needs.
- Controllers are trained entirely inside those fast models and then checked against the original physics simulator.
- One recipe covers a wheeled vehicle crossing soft soil and a tracked vehicle operating a mounted robotic arm.

ORCID iDs Harry Zhang, 0009-0003-1903-274X; Dan Negrut, 0000-0003-1565-2784.

1 Introduction

Complex robots, e.g., wheeled vehicles crossing deformable terrain, tracked platforms, and articulated manipulators, are governed by multibody dynamics, frictional and deformable contact, and actuator behavior that resist compact analytical description. High-fidelity embodied AI simulators such as Project Chrono [1, 2] resolve these effects and provide a controlled, repeatable environment in which a candidate controller can be evaluated before hardware trials [3]. That fidelity is expensive. A detailed granular-terramechanics or tracked-vehicle scene can run several times slower than real time, while reinforcement learning (RL) consumes millions of environment transitions across repeated training runs. Training a policy directly inside such a simulator is impractical for the scenarios where high fidelity matters most. High-fidelity simulation supplies the physics needed to evaluate a controller, but not the throughput needed to search for one efficiently.

A learned dynamics model is one common remedy for this cost, see, for example, [4], but introducing one does not by itself resolve the difficulty; the open question is what the model should represent. Learning the transition of the complete simulator state is neither necessary nor easy: many internal degrees of freedom need not be represented explicitly for the control task at hand, some quantities are deterministic functions of others, and contact and deformable-terrain states can be extremely high-dimensional. An aggressively simplified surrogate has the opposite failure mode. It runs quickly, but may discard the tire slip, load transfer, or actuator lag on which the controller depends, so that a policy trained in it may fail when transferred to the high-fidelity dynamics. Neither low one-step prediction error nor generic state compression establishes that a surrogate is fit for policy learning. This raises the central question of the paper: what should a neural reduced dynamics (NRD) model retain, receive as input, reconstruct analytically, and omit, so that it is both fast and useful for control?

We argue that a useful NRD model should preserve task-relevant fidelity rather than minimize state dimension for its own sake. It should propagate the recurrent variables on which the task-relevant dynamics depend, treat externally specified commands and regime information as inputs where appropriate, and recover quantities that follow from the propagated state through known kinematic relationships outside the learned transition. Figure 1 shows the resulting pipeline, which is shared by both case studies. Excited trajectories are collected from Chrono; a history-based NRD model built on a causal transformer [5] is trained on a task-specific reduced state and selected by open-loop rollout fidelity rather than one-step loss; the selected model is frozen and vectorized so that a policy can be trained against thousands of copies with proximal policy optimization (PPO) [6]; and the trained policy is returned to the full Chrono system for closed-loop evaluation. Here *reduced* denotes preserved task-relevant fidelity, not minimal dimensionality, and closed-loop transfer to Chrono is the decisive criterion, while the NRD model's supervised error and the policy's reward inside it serve as intermediate evidence.

We instantiate this framework in two complementary case studies. Case Study I builds a terrain-conditioned NRD model for the HMMWV. Its reduced state keeps body dynamics, tire normal loads, and wheel speeds, and an explicit terrain context tells the model whether the ground is rigid or deformable. A single generalist policy is trained inside that model. Back in Chrono, that policy attains the lowest median and mean closed-loop tracking error among the policies compared here, on rigid flat terrain, on CRM deformable soil [7, 8], and on bumpy rigid terrain it never saw during training. Each single-terrain specialist, in contrast, degrades sharply off its training terrain. Case Study II applies the same pipeline to two substantially different abstractions on one tracked-vehicle platform: a three-dimensional planar state for tracked-vehicle goal reaching and an eight-dimensional joint-space state for arm reaching, with vehicle pose integrated and end-effector position recovered through forward kinematics outside the NRD model. Policies trained entirely inside these frozen NRD models transfer to Chrono, reaching 100 of 100 vehicle goals within 0.75 m and 97 of 100 arm goals within 0.05 m, with zero contacts or joint-limit violations across the arm rollouts. Across both cases, the batched NRD models achieve measured aggregate simulated-time throughput ratios of up to 58,400× under the reported GPU-batched surrogate and single-process Chrono configurations, making iterative on-policy RL practical.

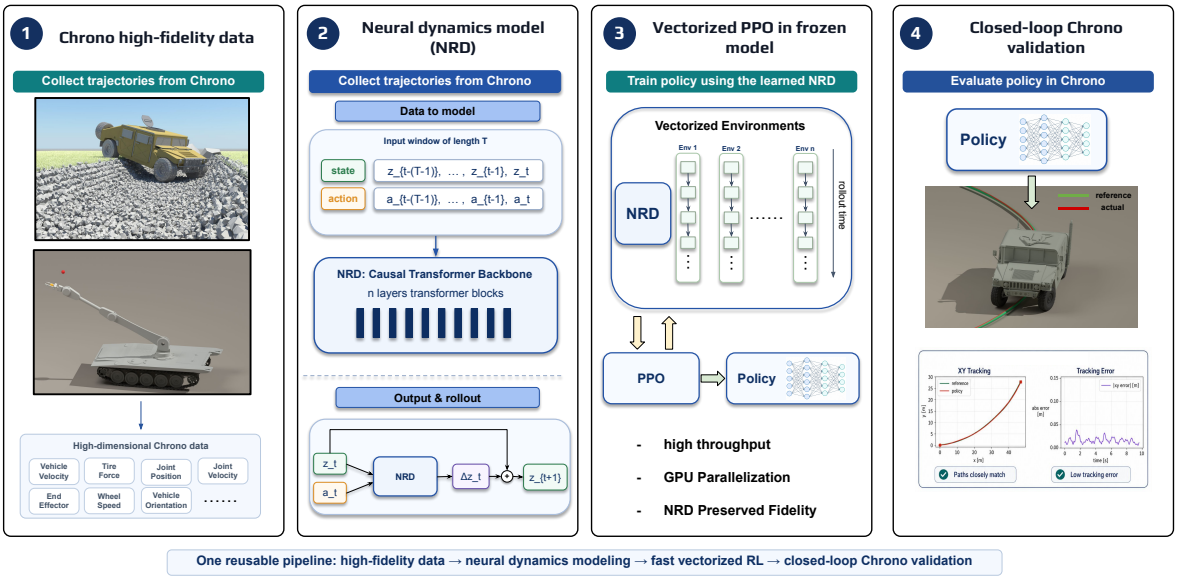


Fig. 1 Overview of the neural reduced dynamics framework. A single reusable pipeline turns expensive high-fidelity physics into scalable policy learning: (1) high-fidelity Chrono simulation generates state–action trajectories for complex robots (a terrain-traversing HMMWV and a tracked vehicle with an articulated arm), yielding high-dimensional signals such as vehicle velocity, tire force, joint position/velocity, end-effector position, wheel speed, and vehicle orientation; (2) a causal-transformer neural reduced dynamics (NRD) model learns next-step reduced dynamics from a window of state–action history; (3) vectorized PPO trains a control policy entirely inside the frozen, high-throughput NRD model; and (4) the resulting policy is validated closed-loop back in high-fidelity Chrono, where it is evaluated closed-loop on the original control task

Our contribution is methodological. The paper studies how control-relevant state, applied actions, regime context, and known analytical structure should be organized within a learned surrogate, and evaluates that organization through downstream policy transfer. The causal transformer and PPO implementation provide just one realization of the framework rather than the central contribution. Specifically, the paper makes the following contributions.

- **Task-specific neural reduced-dynamics framework.** A reusable "Chrono-data"–to–"policy" pipeline that separates recurrent state, applied actions, optional context, and analytically derived quantities; selects models by held-out rollout fidelity; freezes and vectorizes them for policy training; and validates the resulting policies through closed-loop Chrono transfer.
- **Terrain-conditioned HMMWV dynamics and control.** A fifteen-dimensional terramechanics abstraction with a rigid/CRM context code, supported by reduced-state and context ablations, whose generalist policy attains the lowest median and mean tracking error among the compared policies on flat, CRM deformable, and zero-shot bumpy terrain.
- **Cross-task applicability of task-specific abstractions.** Distinct three-dimensional tracked-vehicle and eight-dimensional arm abstractions on one vehicle-plus-arm platform, producing policies that reach 100/100 and 97/100 Chrono goals, respectively.

2 Background and Related Work

2.1 Learned Dynamics and Neural Surrogates for Control

Learned transition models build off predicted experience rather than costly interaction with a physical system or a high-fidelity simulator. Probabilistic ensembles with trajectory sampling (PETS)

pair learned dynamics with sampling-based model-predictive control, propagating model uncertainty into planning to obtain sample-efficient control [9]. The accompanying risk is autoregressive model bias: as a model is advanced on its own predictions, error compounds and an optimizer begins to exploit the model rather than the system it stands for. Model-based policy optimization (MBPO) manages this by branching short synthetic rollouts from real states, bounding how far the model is trusted [10]. The learned model here occupies a more demanding role: it is the entire environment in which the policy is trained, advanced over complete episodes, so its behavior under sustained recursion is the property that matters. That role dictates the evidence reported below: one-step accuracy, autoregressive open-loop rollout against recorded Chrono trajectories, and closed-loop policy transfer back to Chrono.

Latent world models address the same cost problem from a different starting point, learning compact dynamics directly from high-dimensional observations and optimizing behavior inside the learned model. World Models train a controller inside an imagined environment [11], PlaNet learns a latent state-space model from images and plans through imagined trajectories [12], and Dreamer learns actor and value functions from latent imagination rather than planning in the original environment [13]; recent work pretrains such a model in simulation and adapts only its dynamics component with real observations [14]. Common to this category is that the representation itself must be discovered: raw observations do not name physical quantities, so an encoder is learned jointly with the dynamics. For this work, we start from the opposite situation. The simulator already exposes variables with physical identity—body velocities, tire normal forces, wheel speeds, joint angles—so the central representation problem is physical-variable selection and validation rather than perceptual encoding: which variables must be propagated recurrently, which can be supplied as inputs, and which follow from known kinematics. We freeze the selected model rather than adapt it, and the transfer we evaluate is sim-to-sim.

The closest contemporary comparisons are neural surrogates built for robot dynamics. Neural Robot Dynamics learns robot-centric dynamics with learned contact-solver backends and uses the result as a neural simulation engine for policy training [15]. Neighboring efforts are shaped by their downstream use: smooth neural surrogates regularize sensitivity so that contact derivatives remain usable for gradient-based legged MPC [16]; on-the-fly adaptation updates part of an incremental dynamics model online under changing conditions [17]; and contact-aware neural dynamics predicts upcoming contacts and refines them with real tactile data [18]. The framework developed here learns an offline, task-specific reduced transition rather than a general contact solver, and freezes and vectorizes it for model-free policy optimization rather than differentiating through it or adapting it online. What separates these approaches is which state is retained, how the model is used during rollout, and what evidence is taken to show that it suffices for that use.

2.2 Reduced-Order and Structured Dynamics Models

Projection-based reduced-order modeling constructs a low-dimensional subspace offline and evolves the system inside that subspace online, trading a controlled loss of fidelity for a large reduction in cost. Benner et al. survey this offline/online structure across parametric dynamical systems [19]. Simulation-oriented reduction continues to build coordinates with physical meaning: FreeForm derives a mesh-free, material-aware deformation basis from particle-based skinning eigenmodes to accelerate hyperelastic simulation [20]. Hybrid constructions retain efficient reduced mechanics and learn only the part the subspace fails to resolve, as when handle-based subspace dynamics are augmented with learned nonlinear corrections that restore contact-induced deformation in the same reduced coordinates [21]. The same idea appears in vehicle dynamics, where manifold-based linear flow-map surrogates and a learned correction term repair a simplified vehicle model from sparse data without assuming a prescribed form for the correction [22]; that work is assessed by the predictive accuracy of the corrected model, whereas the question here is whether a learned NRD model is sufficient for downstream policy optimization. The principle shared with this paper is that reduction is a decision about which coordinates, which dynamics, and which outputs are preserved, rather than the act of placing a smaller black-box network around the full simulator state.

Where the reduced coordinates come from separates two design attitudes. Linearly recurrent autoencoder networks learn nonlinear latent coordinates jointly with an approximately linear recurrence in those coordinates, so the representation is discovered from data under a structural prior on the dynamics [23]. The projection used here is instead chosen from simulator knowledge and from what the downstream controller requires. Three consequences follow. The retained channels keep their physical identity, so the abstraction can be inspected and revised in physical terms. The model does not decode or reconstruct the complete Chrono state. Quantities fixed by known relationships, including global vehicle pose and arm end-effector position, are recovered analytically outside the network. In this paper, *reduced* therefore means discarding information that is task-irrelevant, or that the transition does not depend on and can be recovered analytically, while preserving the unresolved dynamics the controller depends on. It does not mean minimizing latent dimension or reproducing every simulator output.

2.3 High-Fidelity Simulation as the Physical Reference

All high-fidelity data in this paper are produced with Project Chrono, an open-source multi-physics engine covering multibody dynamics, frictional contact, finite elements, and fluid–solid interaction [2]. Two of its modules account for the two platforms studied here. Chrono::Vehicle provides template-based ground-vehicle models in which suspension, steering, driveline, powertrain, and tire subsystems—and, for tracked platforms, sprocket, idler, roadwheel, and track-shoe assemblies—are instantiated as separate components rather than approximated by a lumped chassis model [3]. Terrain interaction is modeled at three levels of fidelity: rigid terrain; the semi-empirical Soil Contact Model (SCM), which supports deformable terrain for wheeled and tracked vehicles at interactive rates [24]; and the Continuum Representation Model (CRM), an SPH-based continuum granular model coupled to the multibody system through the FSI interface [8]. Case Study I uses rigid terrain and granular CRM soil, with CRM defining the deformable-soil regime.

This combination is the reason Chrono is used as the reference simulator here. Engines developed primarily for robot learning—MuJoCo [25, 26], Isaac Lab [27], Brax [28], and Genesis [29]—are designed around high-throughput vectorized rollouts, and the more recent ones add particle-based deformable and fluid solvers, but they do not supply the vehicle subsystem templates, powertrain and tire models, or vehicle-oriented terramechanics that the wheeled and tracked case studies require. Chrono resolves that physics at a throughput cost measured in Appendix A. It is accordingly used in this work as the source of training trajectories and as the reference in which transferred policies are evaluated, while the throughput needed for policy search comes from the NRD models of Sec. 3.

2.4 Terrain- and Context-Conditioned Dynamics

Off-road motion prediction improves when the learned dynamics are conditioned on terrain. A terrain-aware kinodynamic model conditions its prediction on terrain-related information available from proprioceptive and exteroceptive observations, and supplies the result to a sampling-based predictive controller [30]. Recent work sharpens this line by propagating dynamical uncertainty several steps ahead for risk-aware speed planning and by augmenting a nominal vehicle model with an online Gaussian-process residual on uneven ground [31, 32]. In each case the learned component acts online, inside the control loop and continually reconciled with incoming data, whereas the NRD models developed here are learned offline from high-fidelity trajectories and then frozen, serving as the environment in which a policy is trained rather than as a predictor within a planner. A complementary group of methods dispenses with a learned dynamics model altogether, embedding an analytical rigid-body model and an explicit rollover constraint in GPU-parallel MPC [33] or training an end-to-end driving policy directly inside the original simulator [34]. Several of these studies report results on full-scale vehicles [31, 33, 34], whereas the transfer demonstrated here returns the policy from the NRD model to the high-fidelity simulator, in exchange for a controlled test of whether a chosen abstraction suffices for control.

Conditioning a learned transition on a dynamics context is the general form of this idea. Context-aware dynamics models infer a latent context from a short history of recent transitions so that a single

learned model adapts its predictions across systems or operating conditions [35]. The HMMWV study uses the same general conditional-modeling principle with a deliberately simpler context. The context is a known two-class label for the interaction regime, rigid or deformable CRM soil, rather than a latent variable inferred online; it enters the dynamics model and not the policy observation; and it performs neither terrain recognition nor online identification of terrain parameters. Bumpy rigid terrain accordingly retains the rigid code, since the geometry changes while the underlying contact regime does not. Its purpose is to resolve transition ambiguity: similar reduced states under similar commands evolve differently under rigid and deformable interaction.

Taken together, these lines of work fix the ingredients of the framework developed in this paper. Learned dynamics establish that predicted experience can carry the cost of control optimization, subject to how far the model is trusted. Reduced-order modeling establishes that the choice of retained coordinates and outputs, not the size of the network, determines which fidelity survives reduction. High-fidelity simulation supplies the physical reference and the setting in which a controller is ultimately judged. Context conditioning shows how physically distinct operating regimes can be disambiguated when the reduced transition alone is ambiguous. The combination addressed in this work is a task-designed physical reduction with analytically reconstructed kinematics, a frozen model used as a fully vectorized policy-learning environment over complete episodes, and validation by returning the trained policy to the simulator that produced its training data. Off-road and robot-dynamics studies generally develop a controller or a dynamics model for one platform; the question examined here is whether deliberately selected, task-specific abstractions can each serve as massively parallel policy-training environments across distinct robot embodiments and control tasks.

3 Neural Reduced Dynamics Framework

Figure 1 shows the pipeline shared by both case studies. High-fidelity Chrono trajectories are distilled into a task-specific neural reduced dynamics (NRD) model; that model is frozen and replicated into a vectorized environment in which a control policy is trained; and the trained policy is returned to the full Chrono system for closed-loop validation. The NRD model predicts a compact, task-specific reduced state rather than the complete simulator state, and the central design decision is not the neural architecture but how the information available from the simulator is assigned among four roles: the recurrent state the model learns to propagate, the action applied to the modeled system, an optional context input naming the operating regime, which is needed when the same reduced state and action would evolve differently from one regime to the next, and quantities recovered analytically outside the model. This section defines that machinery; Sec. 4 and Sec. 5 instantiate it, and Sec. 6 interprets what the two instantiations teach.

3.1 Reduced-Dynamics Formulation

Let x_t denote the full state of the high-fidelity Chrono system, which advances under an applied command a_t as

$$x_{t+1} = F_{\mathrm{HF}}(x_t, a_t), \tag{1}$$

where F_{HF} is a single Chrono step and is expensive to evaluate. The framework does not approximate F_{HF} in full. It instead models the evolution of a task-specific *reduced state* $z_t = P(x_t)$, a low-dimensional projection P of the simulator state chosen per task (Sec. 3.2). The NRD model f_θ predicts the reduced-state increment from a finite window of the current step and the k preceding ones, using the reduced states $z_{t-k:t}$, the applied actions $a_{t-k:t}$, and an optional context input $c_{t-k:t}$ that labels the operating regime,

$$\begin{aligned} \Delta\hat{z}_t &= f_\theta(z_{t-k:t},\, a_{t-k:t},\, c_{t-k:t})\,, \\ \hat{z}_{t+1} &= z_t + \Delta\hat{z}_t, \end{aligned} \tag{2}$$

where a hat marks a model-predicted quantity, so the predicted increment $\Delta\hat{z}_t$ and next state $\hat{z}_{t+1}$ are distinguished from the ground-truth reduced state z_t. The model thus learns a residual update, and the history supplies temporal information about actuator, contact, and terrain effects that the instantaneous reduced state does not resolve. Equation (2) describes teacher-forced one-step prediction during supervised training; in

open-loop rollout and policy learning, the ground-truth state history is replaced recursively by the model-predicted history.

This formulation partitions the information available from the simulator into four roles. The reduced state z_t holds the recurrent quantities the model must learn and propagate: the coordinates on which the transition depends and whose evolution is uncertain or contact-dependent. The action a_t is the command actually applied to the modeled system. The context c_t is an optional regime input, used only when the same reduced state and action evolve differently under physically distinct conditions. Finally, quantities that are deterministic functions of the predicted state history, written $y_t = G(\hat{z}_{\le t})$, are recovered analytically *outside* the model rather than assigned learned output channels. These are coordinates on which the transition does not depend, such as global vehicle pose (x, y, ψ), integrated from the predicted body velocities, and known kinematic outputs, such as the arm end-effector, obtained by forward kinematics on the predicted joints.

3.2 Abstraction and Architecture

The projection P follows a single rule: the reduced state retains the variables required to propagate the task-relevant behavior, that is, the coordinates on which the transition depends, and excludes coordinates to which the dynamics are invariant, such as global pose on flat ground, and deterministic geometric outputs that can be reconstructed outside the learned model. A variable is *control-relevant* in this sense when the controller would act differently if it took a different value, and the operational test is deletion: a channel earns its place only if removing it degrades rollout fidelity or closed-loop performance (Appendix E). Applying this rule, vehicle pose, on which the planar response does not depend, is integrated from predicted body velocities rather than predicted directly, whereas the arm's joint positions, on which its response does depend, are retained in the state even though they too could be integrated (Sec. 5.2); the arm end-effector is recovered through forward kinematics rather than learned; and a terrain regime that changes the transition is supplied as context rather than inferred. The two case studies show that the rule resolves to substantially different states—a fifteen-dimensional terramechanics state for the HMMWV, a three-dimensional planar state for the tracked vehicle, an eight-dimensional joint-space state for the arm—without changing the principle. The dominant physics differs across the three tasks, but the design principle is the same: the model propagates only the coordinates on which the dynamics depend, and everything else is supplied as an input or reconstructed from known relationships.

All three NRD models share one architecture family: a GPT-style causal transformer over a sequence of tokens, each concatenating the reduced state, the action, and (when present) the context code at one time step. Causal self-attention processes a finite window of this history, and a linear head predicts the reduced-state increment $\Delta\hat{z}_t$; advancing $\hat{z}_{t+1} = \hat{z}_t + \Delta\hat{z}_t$ produces the open-loop rollout. Attention over history lets the model represent actuator- and contact-mediated effects that the instantaneous reduced state omits and that a memoryless one-step map cannot capture. Model depth, width, and context length are selected separately for each task from preliminary validation and computational cost (Sec. 4, Sec. 5), so the case studies share this architecture family but not an identical network capacity or state definition.

3.3 Training and Model Selection

Each NRD model is trained by supervised learning on Chrono trajectories, split at the episode level so no trajectory is shared between training and evaluation, with state and action channels normalized to comparable scales. The target is the one-step reduced-state increment, trained under a per-channel weighted regression loss

$$\mathcal{L}_{\text{step}} = \sum_j w_j \, \rho(\Delta z_{t,j} - \Delta\hat{z}_{t,j}) \,, \tag{3}$$

where ρ is a squared or Huber penalty and the weights w_j balance channels whose increments differ in scale. For the tracked-vehicle and arm models the normalized channels are already comparable, so this reduces to an unweighted mean-squared error; the HMMWV model pools two terrain regimes whose increment scales differ sharply, and there a per-channel domain rebalancing and the robust Huber form keep a few high-variance channels from dominating the fit (Sec. 4.3).

Training minimizes one-step error, but one-step error is not the quantity that governs downstream control. A model with low one-step loss can still drift when its own predictions are fed back over a long horizon, and it is that autoregressive behavior the policy trains against. We nevertheless keep the training loss one-step rather than backpropagating through multi-step autoregressive rollouts, since gradients through a chain of the model's own predictions are far noisier and less stable to optimize; the horizon is instead accounted for at model selection. We therefore track two signals: a teacher-forced one-step error, and an autoregressive open-loop rollout in which the model is advanced on recorded actions and compared to the Chrono trajectory over a horizon. Every deployed NRD model in this paper is the checkpoint that minimizes held-out open-loop rollout error rather than the checkpoint of lowest one-step loss. The two signals can rank checkpoints differently, so the choice of selection metric is consequential (Sec. 4.3, Sec. 5.3).

3.4 Policy Learning and Validation

Once an NRD model is selected, its parameters θ are frozen and it becomes the environment in which a control policy is trained. Thousands of copies of the frozen model are batched on a single GPU and advanced in parallel, and only the policy parameters are optimized while the learned dynamics stay fixed. During training the policy never receives the full simulator state; its observation is assembled from the model's own predicted reduced state $\hat{z}_t$ and the derived quantities $G(\hat{z}_t)$, and the task reward is evaluated from the same rollout. The frozen model advances that prediction under the resulting action,

$$\begin{aligned} o_t &= h(\hat{z}_t,\, G(\hat{z}_t))\,, \qquad a_t = \pi_\phi(o_t), \\ \hat{z}_{t+1} &= \hat{z}_t + f_\theta(\hat{z}_{t-k:t},\, a_{t-k:t},\, c_{t-k:t})\,, \end{aligned} \tag{4}$$

where h is the task-specific observation map (absorbing the goal or reference, any history, and normalization), and a_t is the command finally applied to the dynamics, with deterministic command processing folded into the policy interface. The throughput of the frozen model—orders of magnitude above Chrono—is what makes on-policy reinforcement learning practical (Sec. 6.4). Both case studies train with vectorized PPO, differing only in the observation, reward, and action interface each task requires.

Validation proceeds at three levels of increasing stringency: (1) one-step accuracy, which checks the learned transition locally; (2) open-loop rollout, which checks whether repeated NRD predictions stay close to recorded Chrono trajectories; and (3) closed-loop transfer to Chrono, which tests whether a policy trained inside the frozen NRD model still works in the high-fidelity simulator. Training and this final evaluation are the same closed loop with the same policy interface (Eq. 4), differing only in the source of the state transition. In Chrono the trained policy is inserted unchanged and one step reads

$$\begin{aligned} o_t &= h(z_t,\, G(z_t))\,, \qquad a_t = \pi_\phi(o_t), \\ x_{t+1} &= F_{\mathrm{HF}}(x_t, a_t), \qquad z_{t+1} = P(x_{t+1}), \end{aligned} \tag{5}$$

with $z_t = P(x_t)$: the full high-fidelity state is advanced by F_{HF} and projected back through P to rebuild the same observation. From the policy's perspective only the dynamics provider changes: the frozen NRD model during training and Chrono during evaluation. The policy weights, observation construction, action processing, and safety interface carry over untouched. Closed-loop transfer is decisive because low supervised error and high reward inside the NRD model do not by themselves rule out a policy that exploits the model's approximation error; only evaluation in Chrono confirms that the abstraction preserved the dynamics control requires. That test is decisive for the abstraction, not for real-world performance: whether these policies transfer to hardware is a question of sim-to-real transfer, which lies outside the scope of this paper (Sec. 7). Section 6 returns to this hierarchy and to what the two case studies reveal about choosing the reduced state.

4 Case Study I: Terrain-Aware HMMWV Dynamics and Control

This case is the strongest evidence that off-road control requires a terrain-aware abstraction. A fixed HMMWV operates across two physically distinct terrain regimes—flat rigid terrain and CRM deformable soil—with unseen bumpy rigid

heightmaps reserved as a strictly held-out zero-shot robustness test. Vehicle and wheel parameters are kept the same throughout, so terrain interaction is the only source of domain shift. All datasets, trained models, experiment configurations, and training results reported in this paper, for both case studies, are available on the project page [36].

4.1 System and Task

The high-fidelity source is the Chrono `HMMWV_Full` model, held fixed across every dataset: SMC contact, a `SHAFTS` engine with an `AUTOMATIC_SHAFTS` transmission, all-wheel drive, and Pitman-arm steering [3]. The task is trajectory tracking across three terrains—flat rigid terrain, bumpy rigid heightmaps, and CRM deformable soil—that share this vehicle and powertrain and differ only in the tire–terrain interaction. Because the vehicle configuration is identical everywhere, terrain interaction is the sole source of domain shift.

Data are generated with deliberately excited driver maneuvers rather than a single path-following controller, so the training distribution spans straight-line launch/brake, transient cornering, and combined longitudinal–lateral response. Six maneuver families are shared across regimes sampled over low, medium, and fast speed bands. Each episode discards an initial settling transient (a per-scenario warmup window) before recording at 100 Hz, and all splits are episode-level so no trajectory is shared between training and test.

Three datasets are collected with distinct roles (Table 1). Flat and CRM are used for training, validation, and in-domain testing; bumpy rigid terrain is reserved strictly for final zero-shot out-of-distribution (OOD) evaluation and never enters training, checkpoint selection, normalization, or reward tuning. Bumpy terrain keeps the *rigid* regime code (Sec. 4.3): it perturbs local geometry but not the contact physics.

The three regimes differ substantially in their contact physics and, correspondingly, their simulation cost (Table 2). Flat and bumpy rigid terrain use `TMEASY` tires [37] on an SMC rigid contact surface at friction $\mu = 0.9$, integrated at a 2 ms step. The CRM terrain replaces the rigid surface with a Chrono SPH–FSI deformable-soil model [7, 8]: the four wheels are registered as FSI rigid bodies carrying rigid tires, spindle forces are read from the FSI solver rather than an analytical tire model,

Table 1 HMMWV datasets and their roles. Sizes are episodes and processed 15-D transitions (training / validation). Bumpy rigid terrain keeps the rigid regime code and is held out for zero-shot evaluation only

Dataset	Role	Scale	Code
Flat rigid	Train, val, in-domain test	≈82k episodes (∼300 GiB); 329 M / 81 M transitions	[1, 0]
CRM soil	Train, val, in-domain test	2,000 episodes; 2.28 M / 0.60 M transitions	[0, 1]
Bumpy rigid	Zero-shot OOD test only	20 episodes are selected for chrono closed-loop policy evaluation	[1, 0]

Table 2 Simulation settings by terrain regime. All regimes share the fixed `HMMWV_Full` vehicle (SMC contact, `SHAFTS` engine, automatic transmission, AWD, Pitman-arm steering) and record at 100 Hz; they differ only in the tire–terrain contact model

Setting	Rigid (flat, bumpy)	CRM (soil)
Tire model	`TMEASY` (analytical)	`RIGID_MESH` (FSI)
Terrain / contact	SMC rigid surface	SPH–FSI deformable soil
Tire-force source	analytical tire report	FSI spindle force
Friction / soil	$\mu = 0.9$	coh. 5 kPa, μ_s=0.8, ρ=1700 kg/m^3, E=1 MPa
Integration step	2 ms	0.5 ms
Terrain extent	flat / 500×500 m heightmap (±0.6 m)	150×150×0.25 m box, 0.08 m spacing

and the soil is a cohesive granular medium (density 1700 kg/m^3, cohesion 5 kPa, internal friction 0.8, $E = 1$ MPa). The SPH field uses 0.08 m initial particle spacing over a 150×150×0.25 m box with a moving active domain, and requires a 0.5 ms integration step—4× finer than rigid—on top of a per-step neighbor search, making CRM episodes markedly more expensive to simulate than rigid ones. Appendix A reports the measured throughput of each regime and of the neural surrogate.

The maneuver mix is preserved across regimes but adapted to CRM's finite domain: the CRM set uses shorter 12–18 s episodes and reduced top speed and steering amplitude so that turning trajectories stay inside the 150 m box. Bumpy rigid terrain is generated from a heightmap library (`bumpy_field_*` patches, 500×500 m, height amplitude ±0.6 m) applied to the same rigid contact

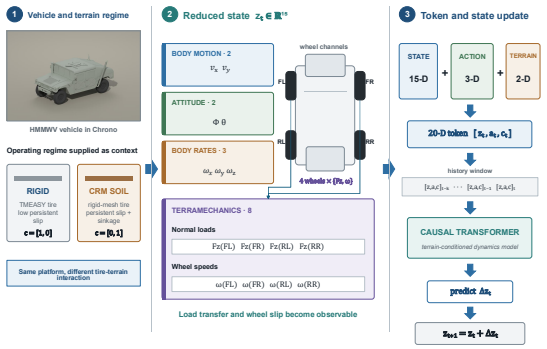


Fig. 2 Control-relevant reduced state for the HMMWV. A fixed vehicle interacts with rigid terrain ($c = [1, 0]$) or CRM deformable soil ($c = [0, 1]$); the 15-D reduced state z_t collects body motion (v_x, v_y), attitude (ϕ, θ), body-frame rates ($\omega_x, \omega_y, \omega_z$), and the eight-channel terramechanics block of tire normal loads $F_z^{(i)}$ and wheel speeds $\omega^{(i)}$ that make slip and load transfer observable. At each step the state, the 3-D action, and the 2-D terrain code form a 20-D token; the causal transformer NRD model predicts the increment Δz_t and advances $z_{t+1} = z_t + \Delta z_t$

model. Its held-out test set covers the same six maneuver families, and every closed-loop evaluation is run on the exact heightmap its reference trajectory was recorded on, so the desired path and the terrain beneath it always correspond.

4.2 Reduced State Abstraction

The reduced-state design uses the following 15-dimensional state:

$$z = \begin{bmatrix} v_x,\ v_y, & \text{(longitudinal/lateral velocity)} \\ \phi,\ \theta, & \text{(roll, pitch)} \\ \omega_x,\ \omega_y,\ \omega_z, & \text{(body-frame roll/pitch/yaw rates)} \\ F_z^{\mathrm{fl}},\ F_z^{\mathrm{fr}},\ F_z^{\mathrm{rl}},\ F_z^{\mathrm{rr}}, & \text{(tire normal forces)} \\ \omega^{\mathrm{fl}},\ \omega^{\mathrm{fr}},\ \omega^{\mathrm{rl}},\ \omega^{\mathrm{rr}} & \text{(wheel angular velocities)} \end{bmatrix}. \tag{6}$$

The action is $a = [\,\text{steering},\ \text{throttle},\ \text{braking}\,]$, the three driver commands. The first seven channels—planar body velocities v_x, v_y, attitude ϕ, θ, and the roll/pitch/yaw rates—are the body-motion base. The remaining eight are the terramechanics block: the four tire normal (vertical) loads $F_z^{(i)}$ and the four wheel angular velocities $\omega^{(i)}$, one per wheel $i \in \{\mathrm{fl}, \mathrm{fr}, \mathrm{rl}, \mathrm{rr}\}$.

The eight terramechanics channels make the reduced state responsive to terrain interaction by exposing load redistribution and wheel–body motion mismatch. The normal loads $F_z^{(i)}$ carry the load transfer produced by acceleration, braking, and cornering, and on soft or uneven terrain they additionally carry information about the contact-load redistribution induced by deformable or uneven ground that a rigid-terrain model never observes. The wheel speeds $\omega^{(i)}$ play the analogous role for the drive axis: together with the body velocity they define an illustrative per-wheel longitudinal-slip proxy,

$$s^{(i)} = \frac{\omega^{(i)} r - v_x}{\max(|v_x|,\ \epsilon)}, \tag{7}$$

where r is the wheel rolling radius. On rigid terrain the wheel rolls almost freely, $v_x \approx \omega^{(i)} r$ and $s^{(i)} \approx 0$; on deformable CRM soil the same drive torque produces persistent slip of order 25% together with sinkage. Slip is thus the primary quantity that separates the two regimes, and Eq. (7) makes explicit that it is representable only if the reduced state carries *both* the wheel speed and the body velocity. Absolute position (x, y) and heading ψ are excluded from z, since the vehicle response on the training terrains does not depend on them, and are reconstructed during rollout by integrating the predicted body velocities and yaw rate (Sec. 3.1); this keeps the learned state compact and invariant to where and in which direction an episode starts.

We adopt this 15-D state as a deliberate, compact design choice: the body-motion base captures vehicle handling, and the four normal loads and four wheel speeds add exactly the terramechanical quantities—load transfer and wheel slip—through which terrain shapes the response. The state-abstraction ablation of Appendix E confirms this choice: dropping the four normal forces and four wheel speeds from the state and prediction target costs little on rigid terrain but degrades open-loop CRM rollout error by ∼49%.

4.3 Terrain-Conditioned NRD Model

This subsection defines the terrain-conditioned dynamics model at the core of the HMMWV case. Terrain conditioning is needed because the same reduced state–action pair evolves differently on rigid terrain and on CRM deformable soil. Under comparable driving conditions, CRM exhibits substantially larger and more persistent wheel slip as well as a noisier state-transition process. Naively mixing the rigid and CRM training data therefore would average this domain difference and degrade

the model performance. We resolve this ambiguity with an explicit two-class terrain code: rigid is $c = [1, 0]$ and CRM is $c = [0, 1]$, repeated at every step and concatenated with the state and action so that each token grows from 18 to 20 dimensions. Concatenating a small one-hot context code is a standard, low-cost mechanism that lets one shared backbone reuse the common vehicle-dynamics representation while specializing the domain shift between rigid and CRM transitions. Removing the terrain key (Appendix E) leaves one-step loss almost unchanged but more than doubles open-loop rollout error on flat terrain, confirming that its value shows up in compounding rollout error rather than one-step prediction.

Architecture.

The NRD model f_θ instantiates the shared causal-transformer backbone of Sec. 3.2 for the HMMWV. Each input token concatenates the 15-D reduced state, the 3-D action, and the 2-D terrain code—20 channels in total—and the model attends over a context of up to 128 steps to predict the 15-D state increment Δz. The transformer uses 8 layers, 8 attention heads, and a 256-dimensional embedding (no bias, with a 256-unit prediction head), totaling 6.40 M parameters; the model is sized for throughput rather than capacity. This depth is not an arbitrary choice: Appendix C reports an architecture sweep over depth, width, attention heads, and context length, and finds depth the dominant lever, with returns saturating beyond depth 8. All results in this section use the resulting 8-layer backbone.

Training.

Training one model across both terrains is not simply a matter of pooling the data. A flat-terrain specialist does not transfer to CRM at all; sequential finetuning from the flat model onto CRM data degrades the previously learned rigid behavior; and naively mixing the two under a plain mean-squared error is dominated by CRM's tire-force channels, whose per-step increments are far larger than the rigid ones, so both the optimizer and the checkpoint metric chase an essentially aleatoric signal. The corrected procedure draws flat and CRM sub-batches at a fixed 75/25 ratio, attaches the terrain code, normalizes inputs consistently, and replaces the plain loss with a per-channel domain-rebalanced Huber loss

$$\mathcal{L} = \sum_j w_j \,\mathrm{Huber}\Big(\Delta z_j - \widehat{\Delta z}_j\Big), \tag{8}$$

whose weights w_j derive from the combined flat–CRM per-channel scale, so the high-variance CRM force deltas no longer overwhelm the lower-scale but control-critical channels. Checkpoints are selected on long-horizon rollout rather than one-step loss: each epoch, the model is rolled open-loop over held-out flat and CRM episodes and selection minimizes the domain-balanced rollout error

$$S(\theta) = \tfrac{1}{2}\, E_{\mathrm{rigid}}(\theta) + \tfrac{1}{2}\, E_{\mathrm{CRM}}(\theta), \tag{9}$$

where E_d is the open-loop position error on domain d normalized by the distance traveled, which makes the shorter, slower CRM episodes comparable to rigid ones. The per-domain errors are reported alongside the scalar so that neither terrain can dominate selection unnoticed.

The model is optimized with AdamW – $\beta = (0.9, 0.95)$, weight decay 0.1 – at a peak learning rate of 3×10^{-4} decayed to 3×10^{-5} after a 1000-step warmup, with gradients clipped to unit norm. Each of 80 epochs draws 2000 mini-batches of 64 history windows at the 75/25 flat/CRM ratio (1.6×10^5 updates in total); training the full conditioned 8-layer generalist takes about 85 minutes on a single GPU (NVIDIA RTX 4090). The final baseline is trained using the full designated training split: Appendix D shows rollout accuracy keeps improving as more training episodes are added. This observation aligns with transformer-based scaling laws in other robotics applications, which makes it promising for scaling-up and generalizability. Figure 3 shows why checkpoints are selected by rollout rather than loss: the one-step validation loss keeps falling, but on CRM it is dominated by the near-aleatoric tire-force channels and tracks rollout quality poorly, whereas the 10 s open-loop rollout error—the quantity scored by Eq. (9)—identifies the epoch-51 checkpoint used in all downstream evaluation.

4.4 Dynamics Model Evaluation

The purpose of this evaluation is to test whether a single learned dynamics model can serve both terrain regimes, and in doing so to show why training

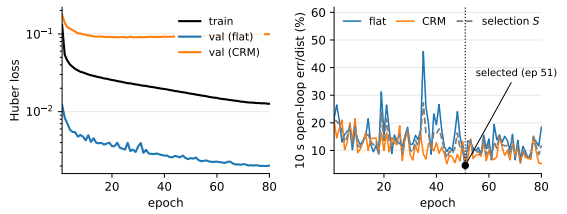


Fig. 3 Training the terrain-conditioned 8-layer generalist (flat+CRM co-training). *Left*: training and per-domain validation one-step Huber loss; the CRM validation loss plateaus higher than the flat case given its larger variation during dynamics transitions. *Right*: 10 s open-loop rollout error normalized by distance for each domain and the domain-balanced selection score S (Eq. 9); the epoch-51 checkpoint that minimizes S is selected

Table 3 Terrain-conditioned generalist vs. single-terrain specialists (rigid-only, CRM-only), each at its selected checkpoint. *One-step loss* is the channel-reweighted Huber validation loss; *rollout err/dist* is the 10 s open-loop position error normalized by distance traveled

	One-step loss		10 s rollout err/dist	
Dynamics model	Rigid	CRM	Rigid	CRM
Generalist (75/25)	0.0025	0.094	3.7%	5.4%
Rigid-only	0.0018	1.27[a]	3.8%	194.1%[a]
CRM-only	0.72[a]	0.11	39.6%[a]	4.2%

[a] Zero-shot: specialist evaluated on the terrain it was not trained on.

on either terrain alone is insufficient. The model is judged primarily by long-horizon open-loop rollout error normalized by distance traveled, which is far more informative for downstream policy learning than one-step validation loss. Table 3 compares the co-trained terrain-conditioned generalist against two single-terrain specialists—one trained only on rigid data, one only on CRM—on held-out rigid and CRM episodes.

The result shows why co-training is necessary, and it shows something stronger than a robustness trade-off. Each single-terrain specialist is close to the generalist on its own terrain (rigid-only 3.8%, CRM-only 4.2%) but collapses off it—the rigid-only model reaches 194% error on CRM and the CRM-only model 40% on rigid—so neither is usable across regimes. The generalist remains close to the respective specialist on each in-domain test (3.7% rigid, 5.4% CRM) while avoiding catastrophic cross-domain failure.

These are open-loop rollout errors on held-out episodes; closed-loop transfer of a single policy trained inside this conditioned model is evaluated in Chrono in Sec. 4.5.

4.5 RL Tracking and Terrain-Conditioned Policy Transfer

With the conditioned dynamics model frozen, we train a trajectory-tracking policy entirely with it. The policy is optimized with PPO across 2,048 vectorized copies of the frozen model on a single GPU; the actor and critic are 512–256–128 ELU multilayer perceptrons with empirical observation normalization. At each control step the policy receives the 231-dimensional observation

$$o_t = \begin{bmatrix} z_{\tau-9:\tau} & \text{state history } (10\times 15) \\ a_{\tau-10:\tau-1} & \text{action history}(10\times 3) \\ z_\tau - z_\tau^{\text{ref}} & \text{state error (15)} \\ e_t^{\text{pose}} & \text{body-frame error (3)} \\ p_{t+1:t+10} & \text{reference preview } (10\times 3) \\ a_{t-1} & \text{last action (3)} \end{bmatrix} \in \mathbb{R}^{231}, \tag{10}$$

where $z_{\tau-9:\tau}$ and $a_{\tau-10:\tau-1}$ denote the 10-step reduced-state and action histories, $z_\tau - z_\tau^{\text{ref}}$ is the reduced-state tracking error, e_t^{pose} is the longitudinal/lateral/heading pose error in the body frame, and $p_{t+1:t+10}$ are the upcoming reference poses, each p contains reference (x, y) coordinate and heading angle; all entries are normalized, and the blocks sum to 231. The observation deliberately excludes the terrain code: terrain enters only through the frozen conditioned dynamics that generate the rollout, and implicitly through the tire-force and wheel-speed state channels, so a single control policy is learned across regimes. The policy outputs the three driver commands $a = [\delta, \tau, \beta]$ (steering, throttle, brake), clipped to their physical ranges with the steering channel additionally rate-limited between consecutive steps (to prevent solver failures due to unsmoothed steering change). One command is held for five dynamics steps, giving 20 Hz control over the 100 Hz reduced dynamics (∼9 s episodes). Training draws on 40 reference segments—20 flat and 20 CRM, each a random ∼11 s mid-episode window covering the six maneuver families; at every episode reset the environment randomly samples one reference matched to that environment's terrain and warm-starts the rollout from the segment's initial recorded history. The RL environment queries the frozen dynamics

model with only the last k=16 of its 128-step training context (Sec. 4.3). This is faster: attention cost is quadratic in context length, and the shorter window is what sustains the ≈ 35,000 policy-control steps/s aggregate throughput (≈ 6.8× faster than the ≈ 5,080 steps/s measured at full 128-step context, same GPU and config) used to train all three policies (Appendix A). It comes at no cost to tracking quality: an open-loop truncation sweep shows 300-step pose RMSE flat across the full context range, and policies trained under k=16 and full-context dynamics reach the same median closed-loop tracking error.

Closed-loop *evaluation* instead uses a separate held-out set of 20 rest-start references per terrain, whose segments begin at vehicle rest so Chrono can warm-start each rollout from zero speed at the reference pose. Note that the closed-loop evaluation trajectories are seen by neither the dynamics model nor the RL policy during training.

The per-step reward pairs a Gaussian tracking term with two small action penalties,

$$r_t = \exp\big(-\mathcal{L}_{\text{track}}\big) \;-\; \lambda_a \,\|a_t - a_{t-1}\|^2 \;-\; \lambda_{tb}\,\tau_t\,\beta_t, \tag{11}$$

whose tracking loss aggregates the position, heading, and body-state errors to the reference,

$$\mathcal{L}_{\text{track}} = w_p\left(\frac{e_p}{\sigma_p}\right)^2 + w_\psi\left(\frac{e_\psi}{\sigma_\psi}\right)^2 + \frac{w_s}{|\mathcal{S}|}\sum_{j\in\mathcal{S}}\left(\frac{z_j - z_j^{\text{ref}}}{\sigma_s\,\hat{\sigma}_j}\right)^2. \tag{12}$$

Here e_p and e_ψ are the position and yaw errors to the reference, $\mathcal{S} = \{v_x, v_y, \omega_z\}$ are the tracked body-state channels scaled by their dataset standard deviations $\hat{\sigma}_j$, and the $\tau_t\beta_t$ term discourages simultaneous throttle and brake. We use $(w_p, w_\psi, w_s) = (2.0, 1.6, 0.2)$, $(\sigma_p, \sigma_\psi, \sigma_s) = (2.0\,\text{m}, 0.35\,\text{rad}, 1.0)$, and $(\lambda_a, \lambda_{tb}) = (0.2, 0.05)$; episodes terminate when the position error exceeds 20 m or the roll/pitch bounds are exceeded. The mixture generalist and the two single-terrain specialists used below share this setup and converge well within budget: the mean episode reward rises steeply and plateaus by ∼300 iterations (Fig. 4), so we evaluate the 1000-iteration checkpoint of each.

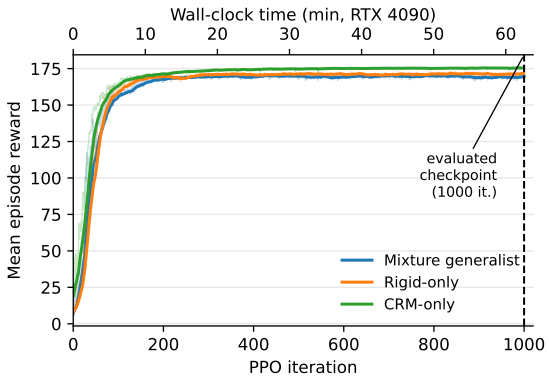


Fig. 4 PPO training convergence for the three tracking policies (mixture generalist and rigid-only / CRM-only specialists), each trained inside the frozen 8-layer conditioned NRD model through 1000 iterations. The top axis converts PPO iteration to wall-clock training time on a single RTX 4090; faint traces are raw per-iteration values; solid curves are exponentially smoothed

Single-terrain specialists.

For comparison we train two specialist policies that follow the *identical* recipe—the same observation (Eq. 10), reward (Eqs. 11–12), PPO architecture, context length k=16, and action clipping—and differ only in the frozen dynamics model and terrain-specific training data. The rigid-only policy is trained with the rigid-only dynamics model (the rigid-only specialist of Sec. 4.4) on 20 flat references with the terrain code held to rigid; the CRM-only policy is trained with the CRM-only dynamics model on 20 CRM references with the code held to CRM. Because the PPO architecture, reward, observation design, and action constraints are held fixed, differences in transfer primarily reflect the learned training dynamics and terrain coverage rather than changes to the controller design.

Three-terrain transfer.

The decisive test is closed-loop transfer of a *single* policy, trained inside the terrain-conditioned dynamics model, back to Chrono across all three regimes. Figure 5 compares the mixture generalist—one policy trained with the rigid/CRM mixed dynamics model (Sec. 4.3)—against the two single-terrain specialists, on held-out rigid flat, CRM, and zero-shot bumpy references under the same action clipping. All nine policy×terrain cells complete 20/20 rollouts with zero early terminations (full per-terrain statistics—median, mean,

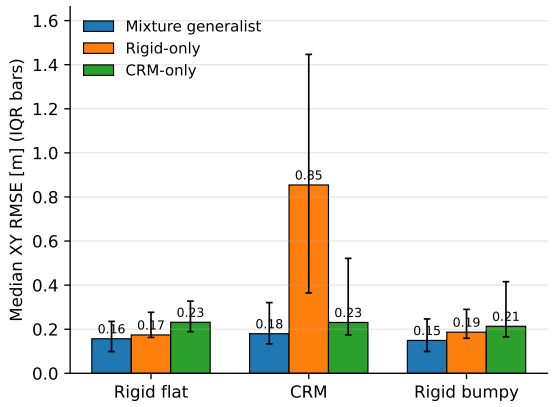


Fig. 5 Closed-loop Chrono policy transfer across the three terrain regimes: one mixture generalist vs. the rigid-only and CRM-only specialists, all trained on the 8-layer dynamics backbone (Sec. 4.3, Appendix C) and evaluated at their 1000-iteration checkpoint under the same action clipping. Bars are the median XY RMSE over 20 held-out references; whiskers span the interquartile range. The mixture generalist achieves the lowest median and mean on all three evaluated terrains, whereas each specialist degrades sharply off its training terrain—most visibly the rigid-only policy on CRM. Bumpy is a strict zero-shot test (rigid code, no bumpy data in training or selection). Full per-terrain statistics are given in Table 11, Appendix B

and Interquartile Range (IQR)—in Table 11, Appendix B). The generalist policy *achieves the lowest mean and median XY RMSE on all three evaluated terrains*. It beats the rigid-only specialist on rigid flat itself (0.157 vs. 0.174 m median) and beats the CRM-only specialist on CRM itself (0.180 vs. 0.231 m median), in addition to its expected wins on bumpy (0.149 vs. 0.187/0.213 m median). Each specialist, by contrast, degrades sharply off its training terrain: the rigid-only policy collapses on CRM (median 0.854 m, mean 1.000 m) and the CRM-only policy is worst on both rigid regimes, see Fig. 9. This is the closed-loop counterpart to the dynamics-level result of Sec. 4.4: with the selected 8-layer backbone, the generalist policy attains lower median and mean tracking error than both specialist policies on every evaluated terrain.

The bumpy column is a strict zero-shot test: no bumpy data enters training, checkpoint selection, normalization, or reward tuning, and bumpy terrain carries the same rigid code as flat terrain (Sec. 4.1). Because the policy never previews the height profile, closed-loop Chrono tracking is the honest metric, and the generalist's bumpy median (0.149 m), the lowest among the compared policies, shows that a policy trained only on flat and CRM data maintains low tracking error on unseen rough rigid terrain—evidence that the conditioned abstraction generalizes rather than memorizes.

5 Case Study II: Reduced Dynamics for a Tracked Vehicle With an Articulated Arm

Case Study I asked whether a suitable reduced state choice can preserve terrain-dependent vehicle dynamics across substantially different contact regimes. Case Study II asks a complementary question about the breadth of the same methodology: can the *same* "high-fidelity-data"–to–"policy" pipeline be reused across different robotic subsystems and control objectives in the absence of a common full-system state? To address this question, we consider a Chrono tracked vehicle with a front-mounted articulated manipulator arm that is subjected to two independent control tasks: planar goal-reaching for the tracked vehicle, and end-effector goal-reaching for the arm. Each task is served by its own task-specific NRD model; each policy is trained entirely inside the corresponding frozen NRD model and then validated in the full Chrono system. The two abstractions differ substantially in dimension and physical meaning, yet both instantiate the causal next-state prediction of Sec. 3. We do *not* present this as simultaneous locomanipulation or coordinated vehicle–arm control; the point is that the right abstraction changes with the subsystem and the task.

5.1 Purpose and Experimental Design

The platform is a stock single-pin tracked vehicle from Chrono::Vehicle (the `M113` model, with 154 rigid bodies and more than 150 degrees of freedom) with a 4-DOF gripper arm rigidly mounted at the front. The two tasks exercise the platform in two modes. In *drive mode*, the vehicle maneuvers to a planar goal while the arm is held at its home configuration as fixed mounted mass, so the NRD model must capture the vehicle's response to driver commands including the mass and center-of-gravity offset the arm contributes.

In *reach mode*, the vehicle is stopped and only the arm moves, so the NRD model must capture the joint response under command-driven actuation, from which end-effector motion is recovered through forward kinematics. Data for both modes are collected in this same mounted-arm Chrono scene, recording at each control step the reduced state and the command applied. In drive mode the welded-arm vehicle is excited by ten randomized maneuver families—steering steps, doublets, sine and chirp sweeps, pivot-like turns, launch–brake, steer-while-braking, multi-steer, coast-down, and stop-and-go—whose amplitudes, frequencies, and durations are drawn per episode to span the vehicle's operating envelope. In reach mode the arm follows a smooth-command sampler that draws a random joint target within the limits and advances q^{cmd} toward it at a bounded rate (resampled about once per second), giving low-jerk actuation rather than white noise; collection is restricted to free-space motion, each episode monitoring arm–ground, arm–vehicle, and arm–self contact by a forward-kinematics signed-distance test (Appendix F) and terminating on contact so no in-collision transitions corrupt the learned distribution. Figure 6 summarizes the platform and the two-task decomposition that organizes the rest of this section. Together, the two modes exercise the complementary capabilities of such a platform: driving the vehicle to a target location and positioning the arm end-effector at a target within its workspace. Developing controllers for such a system directly is computationally demanding—the high-fidelity tracked-vehicle scene carries hundreds of degrees of freedom and rich terramechanics interactions, so every simulation step is costly and full-system policy search is impractical. Routing each mode through the neural reduced dynamics pipeline is what makes the problem tractable: it lets us design and iterate the reinforcement-learning solution efficiently inside compact learned dynamics. Concretely, the high-fidelity vehicle-plus-arm scene steps at only 0.28× real time, whereas the NRD models roll out three to four orders of magnitude faster—up to 58,400× cheaper per simulated second (Appendix A.2).

5.2 Task-Specific State Abstractions and NRD Models

Planar tracked-vehicle abstraction.

The drive-mode reduced state keeps only the planar body motion needed to propagate vehicle pose,

$$z^{\mathrm{veh}} = [\, v_x,\ v_y,\ r\,] \in \mathbb{R}^3, \tag{13}$$

the body-frame longitudinal and lateral velocities and the yaw rate, with action $a^{\mathrm{veh}} = [\,\mathrm{steering},\ \mathrm{throttle},\ \mathrm{braking}\,]$. The global position (x, y) and heading ψ, on which the planar response does not depend, are excluded from the learned state and reconstructed during rollout by integrating the predicted body velocities and yaw rate (Sec. 3.1). We keep only these channels because they suffice for the control task: z^{veh} carries everything the goal-reaching policy needs to be trained in this application. The reduction is aggressive: the Chrono vehicle advances 154 rigid bodies and more than 150 degrees of freedom every step, among them two track chains with their sprocket, idler, road-wheel, and track-shoe assemblies (Sec. 5.1), while the learned model propagates three numbers. None of that mechanism appears in z^{veh}; what survives is the planar motion it produces, which is all the goal-reaching controller acts through.

Articulated-arm abstraction.

The reach-mode reduced state is

$$z^{\mathrm{arm}} = [\, q,\ \dot{q}\,] \in \mathbb{R}^8, \tag{14}$$

the four joint positions q and velocities $\dot{q}$. The action is the four joint command targets $a^{\mathrm{arm}} = q^{\mathrm{cmd}} \in \mathbb{R}^4$, the setpoint the actuators are driven toward over the step.

First, the arm is commanded through q^{cmd} under a per-joint PD impedance control rather than through raw joint torque. PD impedance control is deterministic and yields smoother joint motion than direct torque commands, which gives better-conditioned transitions for the NRD model to learn. Because q^{cmd} is an exogenous setpoint the actuators are regulated toward—an input to the dynamics, not an internal state that evolves on its own—we treat it as the *action* rather than as a state channel. The reaching policy still moves the setpoint by bounded increments Δq^{cmd} for smooth

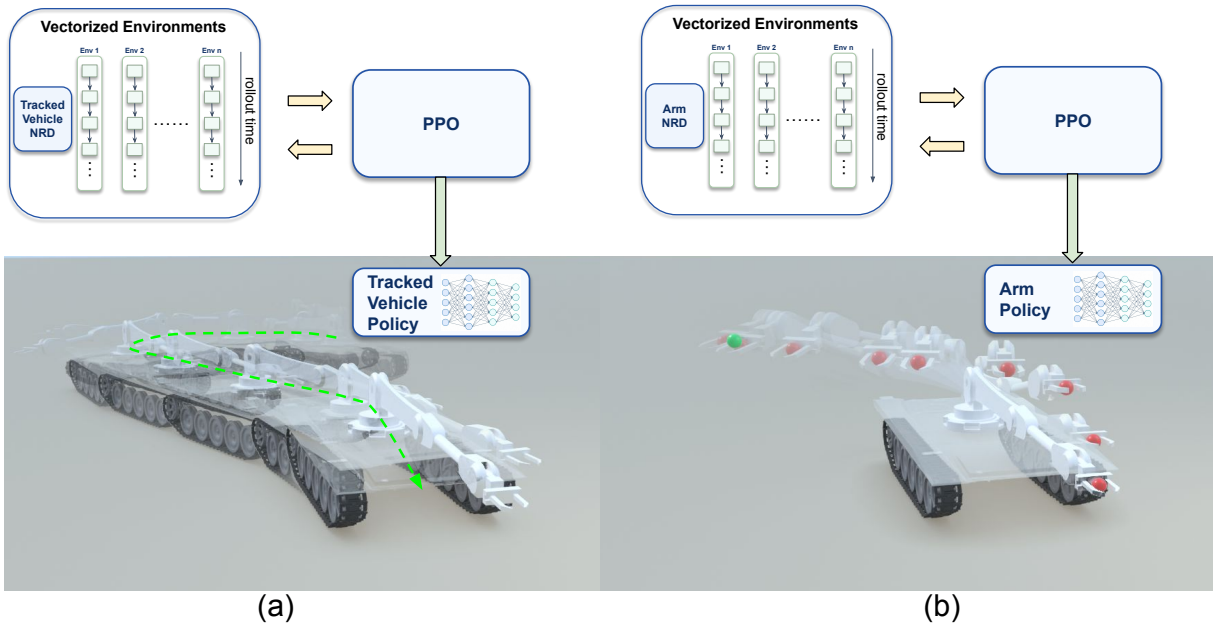


Fig. 6 Platform and two-task decomposition for Case Study II on one vehicle-plus-arm system. (a) Goal reach: the tracked vehicle maneuvers to a planar goal while the front-mounted arm rides at its home configuration. (b) Arm-case overview: with the vehicle held stationary, the 4-DOF arm positions its end-effector at targets across its workspace. Each task has its own reduced state, learned dynamics model, and policy, and each is validated back in high-fidelity Chrono

actuation (Sec. 5.4), but that increment is a property of the policy, not a dimension the dynamics model must carry.

Second, we *exclude* the end-effector from the reduced state and recover its position by forward kinematics on the predicted joints, $p^{\text{ee}} = \text{FK}(q)$. The end-effector is a deterministic geometric function of q, which the model already predicts, so it needs no separate learned channel. Forward kinematics is in any case evaluated every step inside the rollout—the collision-safety shield (Appendix F) already re-derives the end-effector and link geometry from q—so recovering $p^{\text{ee}} = \text{FK}(q)$ costs nothing extra and keeps the reaching target exactly on the kinematics. These two choices leave a compact eight-dimensional reduced state, $[q, \dot{q}]$, that we found accurate enough for open-loop rollout and downstream reaching (Sec. 5.3).

The joint positions are retained rather than integrated from the predicted velocities, as vehicle pose is, because the model should receive the coordinates on which the one-step dynamics depend (Sec. 3.2). The planar vehicle dynamics are invariant to (x, y, ψ), whereas the arm's inertia, gravity load, joint limits, and PD actuator torque are all functions of q: the same driver command applied at different (x, y, ψ) produces the same change in body velocity, whereas the same joint command q^{cmd} at the same $\dot{q}$ produces a different joint response at different q. An ablation with the same architecture, data, and recipe confirms this: a variant that receives only $\dot{q}$ and integrates q from the predicted velocity raises the open-loop end-effector drift from 0.3% to 3.1% err/dist at 0.5 s and from 1.2% to 30% at 2 s, and even a variant that receives q but integrates it the same way is 3–4× worse than the deployed model, since 50 Hz velocity samples miss the intra-step motion of the stiff PD joints.

5.3 Model Training and Accuracy

Both models instantiate the shared causal transformer of Sec. 3.2 over continuous state–action tokens at a 50 Hz ($\Delta t = 0.02$ s) sampling interval with a 16-step (≈ 0.32 s) context, predicting the reduced-state increment and advancing $z_{t+1} = z_t + \Delta z_t$. Their state definitions and model capacities are selected according to their respective tasks (Table 4): the tracked-vehicle model is a compact 3-layer / 4-head / 96-D transformer (0.34 M

Table 4 Task-specific NRD-model configurations. Both instantiate the same causal next-state framework, while their state definitions and model capacities are selected according to their respective tasks. The arm column is the dynamics model used to train the reaching policy

Item	Tracked vehicle	Arm
Physical task	planar goal reach	end-effector reach
Reduced-state dim	3	8
Action dim	3	4
Sampling interval	0.02 s	0.02 s
Context length	16	16
Transformer layers	3	5
Attention heads	4	8
Embedding dim	96	256
Parameters	0.34 M	4.0 M
Training episodes	2,160	15,000
Training transitions	1.41 M	0.76 M
Training epochs	40	80

parameters), while the arm model is a larger 5-layer / 8-head / 256-D transformer (4.0 M parameters). The arm configuration is the exact dynamics model inside which the reaching policy is trained (Sec. 5.4), so every arm number reported here characterizes that same deployed model.

We train each model with the shared objective and causal transformer of Sec. 3 using AdamW with a cosine-decayed learning rate—the tracked-vehicle model over 40 epochs and the arm model over 80 epochs—and freeze the checkpoint of lowest open-loop rollout error against Chrono rather than the lowest one-step validation loss. One-step loss scores each transition in isolation, whereas downstream control depends on the model staying accurate across a rollout, so selecting on the multi-step open-loop error targets that quantity directly. Figure 7 reports both signals: panel (a) the train and validation one-step loss and panel (b) the open-loop rollout error (err/dist) used for selection, with the chosen epoch marked (8 for the tracked vehicle, 76 for the arm). The arm loss converges smoothly—its validation plateaus near 4×10^{-3} with a small train–validation gap and its rollout error falls by an order of magnitude—while the tracked-vehicle one-step loss is noise-limited and nearly flat, so its fidelity is judged from the open-loop rollout error rather than the loss magnitude.

Unlike the HMMWV case, whose heterogeneous terrain dynamics motivated an architecture sensitivity study, the tracked-vehicle and arm models operate on substantially smaller reduced state spaces. We therefore use compact transformer configurations selected from preliminary validation

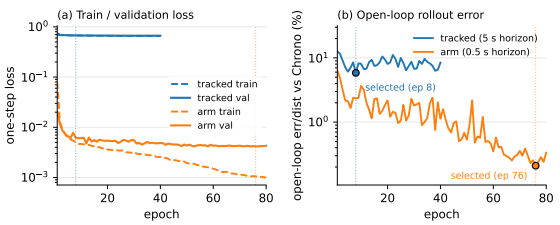


Fig. 7 NRD-model training for both Case Study II models (tracked vehicle, blue; arm, orange), the tracked vehicle over 40 epochs and the arm over 80. (a) train (dashed) and validation (solid) one-step loss, log scale; (b) open-loop rollout error against Chrono ground truth (err/dist, log scale)—the tracked vehicle scored on the 5 s horizon and the arm on the 0.5 s horizon. Each frozen model is selected by *minimum open-loop rollout error* (marker; dotted line)

and do not perform a separate architecture optimization. The purpose of Case Study II is to evaluate whether task-specific reduced abstractions support downstream control, rather than to identify an optimal neural architecture.

Both models are judged by held-out one-step error and, more importantly, by open-loop rollout error normalized by distance traveled (err/dist), the quantity that governs downstream policy learning and transferring back to Chrono. Table 5 reports both. For the tracked vehicle, the model is rolled open-loop on recorded driver commands over held-out episodes and the pose is integrated from the predicted velocities; the 5 s rollout reaches 0.105 m position RMSE at 5.85% err/dist and 1.39° yaw RMSE, so the integrated trajectory stays close over a full maneuver. For the arm, the model is rolled open-loop from a seeded context and the end-effector recovered by forward kinematics from the predicted joints is compared to ground truth; end-effector drift stays near 0.3% err/dist out to 0.5 s and grows to only $\approx 1.2\%$ at 2 s. This fidelity is sufficient to justify training policies inside the frozen models.

5.4 Goal-Reaching Policies in the Frozen NRD Models

Both control tasks are goal-reaching problems, and both are solved the same way: a policy is trained *entirely inside* the corresponding frozen NRD model with vectorized PPO. The two deliberately share one implementation and actor–critic topology (Table 6)—both actor and critic use a $[256, 128, 64]$ ELU multilayer perceptron with empirical observation normalization; `tanh`-squashed actions mapped

Table 5 NRD-model accuracy. Vehicle one-step is the per-channel one-step RMSE of all three reduced states (v_x, v_y in m/s; yaw rate r in rad/s); vehicle open-loop is the 5 s pose-integrated position RMSE / err/dist. Arm one-step is the end-effector position RMSE (forward kinematics on the predicted joints); arm open-loop is the end-effector-drift err/dist at 2 s (0.3% at 0.5 s). All arm numbers are for the deployed 5-layer model used to train the reaching policy (Table 4)

Model	One-step	Open-loop err/dist
Tracked Vehicle NRD model	v_x 0.024, v_y 0.013 m/s; r 0.0069 rad/s	5.85% / 0.105 m XY
Arm NRD model	1.2 mm (ee)	1.2% ee drift

Table 6 Task-specific PPO configurations using a common implementation and actor–critic topology, each trained inside its frozen NRD model; both use `tanh`-squashed actions, GAE ($\lambda = 0.95$), a clip ratio of 0.2, and empirical observation normalization

Setting	Vehicle policy	Arm policy
Parallel environments	2,048	4,096
Control rate	10 Hz	50 Hz
Observation / action dim	11 / 3	26 / 4
Actor/critic hidden	[256, 128, 64]	[256, 128, 64]
Learning rate	3×10^{-4}	1×10^{-4}
Adaptive KL target	0.01	0.005
Discount γ	0.995	0.99
Rollout steps/env	32	64
Epochs / minibatches	5 / 4	3 / 16
Entropy coefficient	0.005	0.001
Success tolerance	0.75 m	0.05 m
PPO iterations	1,500[a]	1,500

[a] The vehicle run is scheduled for 3,000 iterations but converges to $\approx$ 100% NRD-model success within the first $\sim$ 150; the iteration-1,500 checkpoint (long converged) is used for transfer.

into physical bounds; a shaped per-step reward with a terminal success bonus; and clipped PPO with an adaptive-KL learning-rate schedule, GAE, and empirical advantage normalization. Reusing a single policy-learning framework across two very different subsystems is the point of this case study: the tasks share that framework and the actor–critic topology, while the state/action interfaces and task-specific training settings differ.

The two tasks differ in three ways that follow directly from their abstractions. First, the *interface and goal geometry*: the vehicle acts through its driver commands at 10 Hz and reaches a planar location, whose pose is integrated *outside* the learned state from the NRD model's predicted body velocities; the arm acts through joint-command increments at 50 Hz and reaches a three-dimensional end-effector target recovered by forward kinematics from the NRD model's predicted joints, $p^{\text{ee}} = \text{FK}(q)$, so the reaching error follows directly from the rollout. Second, *tolerance and difficulty*: the vehicle must stop within 0.75 m of a goal 20–40 m away, whereas the arm must place its end-effector within 0.05 m while keeping collision- and joint-limit-safe. Third, *scale*: the arm is trained across 4,096 parallel rollouts, the vehicle across 2,048. We detail each task in turn.

5.4.1 Tracked-Vehicle Planar Goal Reaching

The vehicle policy drives the platform from rest at the origin to a goal sampled at radius 20–40 m and arbitrary bearing. At each control step it receives the 11-dimensional observation

$$o_t^{\text{veh}} = \big[\, \hat{v}_x, \hat{v}_y, \hat{r},\ g_x^b, g_y^b, d,\ \sin\theta_e, \cos\theta_e,\ \hat{a}_{t-1} \,\big] \in \mathbb{R}^{11}, \tag{15}$$

the standardized body velocities $(\hat{v}_x, \hat{v}_y, \hat{r})$, the goal in the body frame and its range (g_x^b, g_y^b, d) scaled by 10 m, the sine and cosine of the heading error θ_e, and the normalized previous action. The policy emits $a = [\delta, \tau, \beta]$ (steering, throttle, brake), each `tanh`-squashed and mapped into their physical ranges, with throttle and braking capped at 0.6; one command is held for five dynamics steps, giving 10 Hz control over the 50 Hz NRD model. The per-step reward is a progress term with small regularizing penalties,

$$\begin{aligned} r_t^{\text{veh}} = {} & w_p\,\Delta d - w_\psi|\theta_e| - \lambda_a\|a_t - a_{t-1}\|^2 \\ & - \lambda_{tb}\,\tau_t\beta_t \\ & - \lambda_s\big(\max(|r| - \bar{r}, 0)\big)^2 \\ & - c + b\,\mathbf{1}[\,d < 0.75\,], \end{aligned} \tag{16}$$

where Δd is the reduction in goal distance over the step. We use progress weight w_p=10, heading w_ψ=0.05, action-rate λ_a=0.01, throttle–brake λ_{tb}=0.05, a spin penalty λ_s=0.5 beyond $\bar{r}$=0.4 rad/s, a constant time penalty c=0.1 per step, and a terminal success bonus b=50. The spin and throttle–brake terms suppress the degenerate in-place-rotation and drive-against-brake behaviors that a pure progress reward would otherwise admit.

Training converges quickly: the mean episode reward rises from negative values to a plateau within ~ 250 iterations (Fig. 8(a)), and the episode success rate reaches $\approx 100\%$ within the first ~ 150. At the iteration-1500 checkpoint used for transfer, every NRD-model episode succeeds (100%, mean final distance 0.62 m, no out-of-bounds terminations); the run reaches this in ≈ 10 min of wall-clock on a single GPU.

5.4.2 Arm End-Effector Reaching

The arm policy moves the end-effector to a goal g in the reachable workspace. Goals are drawn in joint space, mapped through forward kinematics, and safety-screened. At each control step the policy receives the 26-dimensional observation

$$o_t^{\text{arm}} = \big[\, q,\ \dot{q},\ q^{\text{cmd}},\ g,\ p^{\text{ee}},\ g - p^{\text{ee}},\ \hat{\kappa},\ \hat{a}_{t-1} \,\big] \in \mathbb{R}^{26}. \tag{17}$$

Here $q, \dot{q} \in \mathbb{R}^4$ are the joint positions and velocities of the reduced state z^{arm} (Eq. 14), standardized by the training statistics. $q^{\text{cmd}} \in \mathbb{R}^4$ is the current absolute joint setpoint, i.e., the actuator command the environment carries from step to step. $g, p^{\text{ee}} \in \mathbb{R}^3$ are the goal and end-effector positions in the arm-base frame, and $g - p^{\text{ee}}$ is the reaching error. $\hat{\kappa} \in \mathbb{R}$ is a clipped clearance signal, a forward-kinematics measure of proximity to collision (Appendix F). $\hat{a}_{t-1} \in \mathbb{R}^4$ is the normalized previous action.

The policy emits a `tanh`-squashed joint-command increment $a_t = \Delta q_t^{\text{cmd}} \in [-\Delta q_{\max}, \Delta q_{\max}]^4$ at 50 Hz, which the environment accumulates into the absolute setpoint q^{cmd} that the NRD model consumes (Eq. 14). Two checks guard this setpoint. First, it is clipped to the joint limits. Second, a geometric safety shield (Appendix F) samples five configurations along the straight joint-space path from the current setpoint to the proposed one, evaluates forward kinematics at each, and tests for contact with the ground, the vehicle, or another arm link; if any sample fails, the increment is discarded and the setpoint is held. The policy thus learns joint-limit and collision avoidance *inside* the NRD rollout, from geometry alone, rather than discovering it later in Chrono.

The reward is an exponential reaching term with an action-rate regularizer and a terminal bonus,

$$r_t^{\text{arm}} = \exp\!\big(-\|g - p^{\text{ee}}\|/\sigma\big) - \lambda_a \|\hat{a}_t - \hat{a}_{t-1}\|^2 + b\,\mathbf{1}[\,\|g - p^{\text{ee}}\| < 0.05\,]. \tag{18}$$

Here σ=0.15 m is the reaching scale, λ_a=0.02 the action-rate weight, and b=150 the success bonus, granted once the end-effector is within the 0.05 m tolerance. The end-effector position is recovered from the NRD model's predicted joints as $p^{\text{ee}} = \text{FK}(q)$, using the same forward kinematics the safety shield already evaluates each step, so the reward reads directly from the rollout at no extra cost.

Reaching is the harder of the two tasks, and its convergence reflects that. The mean episode reward ramps up over roughly 250–600 iterations to a ≈ 150 plateau (Fig. 8(b)), and the episode success rate settles at $\approx 96.9\%$. Safety violations stay rare throughout: at the iteration-1500 checkpoint the shield-flagged collision rate is well under 1% and joint limits are never hit, so the shield shapes behavior without dominating it. The run trains across 4,096 parallel rollouts in $\approx$ 57 min on a single GPU.

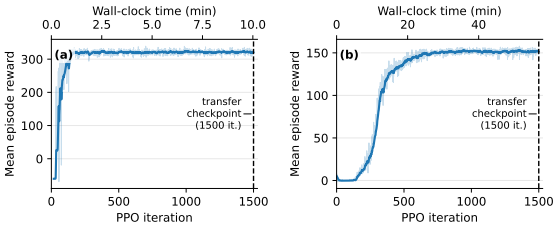


Fig. 8 PPO convergence for the two goal-reaching policies, each trained inside its frozen NRD model: (a) tracked-vehicle planar reaching and (b) arm end-effector reaching. Each panel shows the mean episode reward versus PPO iteration; the top axis converts iteration to that run's wall-clock training time on a single GPU (the arm run is $\approx 6\times$ longer). Faint traces are raw per-iteration values and solid curves are EMA-smoothed. The vehicle reward saturates almost immediately, whereas the tighter 0.05 m arm task ramps up more slowly to its plateau. The dashed line marks the checkpoint used for Chrono transfer

5.5 Closed-Loop Chrono Transfer

Chrono transfer is the decisive test: each policy, trained only inside its frozen NRD model, is deployed in the full high-fidelity vehicle-plus-arm scene. The policy is memoryless in both cases,

so Chrono needs no dynamics context—it reads the true system state, forms the same observation the policy trained on, and applies the resulting command.

Tracked-vehicle transfer.

The vehicle policy is evaluated on a seeded 100-goal stress battery sampled from its training region (radius 20–40 m, full angle $\pm\pi$), with one freshly built Chrono scene per goal. The policy reaches all 100 goals within the 0.75 m tolerance, though this is a coarse criterion: the policy stops on entering the goal region rather than homing onto the goal, so it hugs the tolerance radius (median closest approach 0.691 m, worst 0.748 m). Median time-to-success is 20.2 s (worst 27.7 s) and no episode times out. The driven paths are near-direct—median path efficiency 0.959 (straight-line over driven distance)—and goals behind the vehicle are serviced by clean forward-loop U-turns rather than in-place spins, so the NRD-model policy transfers to the full high-fidelity system across the goal distribution, if to this loose tolerance (Appendix G).

Arm transfer.

The arm policy is evaluated on a seeded 100-goal battery drawn from its trained joint-space workspace, one freshly built Chrono scene per goal after a settle pre-roll. It reaches 97 of 100 goals within the 0.05 m tolerance, with median reached end-effector error 4.2 cm (worst 5.0 cm) and median convergence 0.9 s. All three misses are timeouts with closest approach 6.4, 9.3, and 10.9 cm, not safety failures: every accepted command stays collision- and joint-limit-safe across all 100 Chrono rollouts, with *zero* contacts and *zero* joint-limit violations, so the forward-kinematics safety shield carries from training into the high-fidelity system (Appendix G).

5.6 Case Study Takeaway

Case Study II is not meant to establish that either task is an individually difficult control benchmark. It shows that the proposed NRD methodology is reusable across subsystems with different dominant dynamics and different control objectives. The planar vehicle model retains only the quantities that propagate vehicle pose, whereas the arm model retains only the joint positions and velocities on which the joint response depends, takes the actuator command as its action, and recovers the task-space end-effector by forward kinematics. In both cases, a policy trained entirely inside the corresponding frozen NRD model transfers to the high-fidelity Chrono system—reaching 100% of 100 seeded vehicle goals at the coarse 0.75 m tolerance and 97% of 100 seeded arm goals at the tight 0.05 m tolerance. Together with Case Study I, these results support the paper's claim that neural reduced dynamics should be designed around the control task rather than around a fixed notion of system state—the theme developed next in Sec. 6.

Table 7 Chrono transfer summary over the seeded 100-goal stress batteries (Appendix G). Vehicle: 100/100 within 0.75 m, median closest approach 0.69 m (worst 0.748 m), median time-to-success 20.2 s, median path efficiency 0.959. Arm: 97/100 within 0.05 m (median reached error 4.2 cm), all three misses timeouts, with zero contacts or joint-limit violations. Error scales differ because the two tasks use very different success tolerances

Task	Trials	Success	Median final err
Vehicle goal reaching	100	100%	0.69 m
Arm reaching	100	97%	0.042 m

6 Analysis

The two case studies approach the same question from different directions. Case Study I provides depth: an abstraction can fail when control-relevant terramechanics variables or regime context are omitted. Case Study II provides breadth: the same workflow carries over when the appropriate abstraction changes substantially across tasks and subsystems. Together they support a single account of what a *right* abstraction is—one that follows the task's dominant physics, preserves the fidelity control requires, takes on context only where the reduced dynamics are ambiguous, and earns its keep through throughput—and that is judged, in the end, by whether the policies it trains transfer back into the high-fidelity simulator. Table 8 lays this pattern out across the three tasks.

6.1 Abstraction Follows the Task and Dominant Physics

The three NRD models learn three different states. The terrain-aware HMMWV carries a fifteen-dimensional state that couples body motion and attitude (v_x, v_y, ϕ, θ), the three body-frame angular

Table 8 The same design pattern across three control tasks: each keeps a task-specific reduced state, treats commands and context as inputs, and recovers known kinematics analytically—yet every one yields a policy that transfers into high-fidelity Chrono

Control task	Learned recurrent state	Action / context	Evaluated outside the NRD model	Why this abstraction is needed	Chrono evidence
HMMWV tracking	Body velocities, attitude, and angular rates; per-wheel normal loads and speeds (15-D)	Steering, throttle, braking; rigid/CRM terrain code	Vehicle pose (x, y, ψ); body frame error from reference trajectories	Exposes wheel slip, load transfer, and the domain-dependent tire–terrain response	Lower median and mean tracking error than both single-terrain specialist policies on flat, CRM, and zero-shot bumpy terrain
Tracked vehicle reaching	Planar velocities and yaw rate (3-D)	Steering, throttle, braking	Vehicle pose (x, y, ψ); body frame error from goal point	Skid-steer chassis response in the plane is pose-invariant; pose is integrable	100/100 goals within 0.75 m
Arm reaching	Joint positions and velocities (8-D)	Joint target command q^{cmd}	End-effector $p^{\text{ee}} = \text{FK}(q)$ and contact clearance	Joint response is dynamic and depends on q; task geometry is known analytically	97/100 goals within 0.05 m; zero contacts or joint-limit violations

rates, and an eight-channel terramechanics block of per-wheel normal loads and wheel speeds (Eq. (6)). The tracked vehicle keeps only the three planar quantities its goal-reaching controller acts through—the two body-frame velocities and the yaw rate, $[v_x, v_y, r]$ (Eq. (13)). The arm keeps the eight variables that describe its configuration and how fast it is changing, the joint positions and velocities $[q, \dot{q}]$ (Eq. (14)).

The retained variables differ in kind, not only in count: each controller needs different *physical information*, and the physics that dominates each control problem differs. For the HMMWV the decisive and uncertain physics is the tire–terrain interaction: whether the vehicle holds a reference across flat rigid ground, bumpy rigid ground, and deformable CRM soil is strongly influenced by terrain-dependent load transfer and wheel slip, so the reduced state should expose the tire normal loads and wheel speeds. For the tracked vehicle the task is planar goal reaching within a fixed operating envelope; the controller must anticipate how steering, throttle, and braking produce longitudinal, lateral, and yaw motion through the tracked chassis, and the three retained velocities capture the response required to propagate planar pose. For the arm, reaching is a configuration-space problem: the joints accelerate under actuator torque as they are driven toward a commanded setpoint, so their future motion is set by where the joints are and how fast they are moving—the joint positions and velocities. Because the vehicle is held stationary in reach mode, its motion and tire–terrain variables need not be represented in the arm NRD model.

The lesson is not that a smaller state is better, nor that there is a single correct abstraction to be found. Each abstraction is a compact recurrent state shown by rollout accuracy and closed-loop transfer to retain sufficient control-relevant information for its task, and the dimension is a consequence of that choice rather than the object of it. The right abstraction is task-dependent because the dominant physics is task-dependent.

6.2 Reduction Preserves Task-Relevant Fidelity

Reduction is as much a decision about what *not* to learn as about what to learn. In each case the model propagates only the coordinates on which the transition depends and whose evolution is uncertain or contact-dependent, the part of the dynamics that known relationships (such as FK or coordinate transformation) cannot supply, and leaves the coordinates the transition does not depend on, together with deterministic functions of the predicted state, to be reconstructed analytically during rollout.

The pattern is consistent across the three systems. For both vehicles the global pose is never predicted: the HMMWV's absolute position and heading, and the tracked vehicle's planar (x, y, ψ), are reconstructed by integrating the predicted body velocities and yaw rate outside the learned state

(Sec. 4.2, Sec. 5.2). For the arm, the end-effector position is not a learned channel but is recovered by forward kinematics on the predicted joints, $p^{\text{ee}} = \text{FK}(q)$; the collision clearance the policy observes is computed geometrically by the same kinematics (Appendix F); and the joint target command q^{cmd}, which a low-level PD controller drives the joint angles toward, is supplied to the model as an input (Eq. (14)). The arm's joint positions could likewise be integrated from the predicted $\dot{q}$, but the joint response depends on them, so they are retained in the state (Sec. 5.2); what is reconstructed is decided by whether the transition depends on a quantity, not by whether it is integrable. In every case the learned model carries only the recurrent variables on which the uncertain dynamics depend, and known kinematic and geometric relationships are evaluated analytically rather than represented by additional learned channels.

The learned channels are a projection of the same high-fidelity simulator—trained on its data and checked against it in open- and closed-loop—so they retain fidelity where it is hardest to obtain. The reconstructed quantities remain analytically consistent with the predicted reduced state and require no additional learned output channel; their accuracy relative to Chrono still depends on the accuracy of that predicted state, and the collision-clearance calculation is an approximate geometric screening measure. Reduction therefore targets redundancy rather than fidelity: state that either does not affect the control task, or that the transition does not depend on and can be recovered without learning. The learned channels still carry measurable approximation error, quantified in the one-step and open-loop evaluations of Secs. 4.4 and 5.3; the design goal is not to eliminate that error but to keep it within what the control task tolerates. The resulting model does not reproduce the simulator's full state, and it does not need to.

6.3 Context Is Required When Reduced Dynamics Are Ambiguous

A practical advantage of a learned NRD model over a fixed analytical one is that its data-driven nature lets a single model absorb experience from physically distinct domains: rather than deriving a separate model per regime, we train one backbone on data pooled across regimes and let it represent their shared structure and their differences together. Realizing that advantage rests on two of the design choices this paper isolates—the right abstraction and the right context. The abstraction (Sec. 6.1) gives the regimes a common state in which their shared vehicle dynamics are represented once; the context tells the model, when that shared state is not enough, which regime it is currently in.

Context becomes necessary when the reduced dynamics are *ambiguous*: when the same reduced state and action evolve to different next states under different physical regimes. In the HMMWV case a given body-and-terramechanics state under a given driver command advances differently on rigid ground than on CRM soil, because the two regimes generate different slip and sinkage (Sec. 4.3). Without an explicit label the model must infer the regime implicitly from history, which can blur the learned transition and accumulate error over rollout. A small two-class one-hot code—rigid $[1, 0]$, CRM $[0, 1]$, appended to each input token—resolves the ambiguity, letting one backbone reuse the common vehicle-dynamics representation while specializing the rigid-versus-CRM transition. The context ablation makes the effect concrete: removing the code barely changes one-step loss but more than doubles open-loop error on rigid terrain, from 3.73% to 8.26%, while leaving CRM error approximately unchanged (Appendix E). The history alone therefore does not reliably preserve the rigid/CRM distinction over long rollouts.

6.4 Throughput Makes Policy Learning Practical

Reduction earns its keep not only through accuracy but through speed, because the NRD model replaces Chrono as the environment the policy trains in. A PPO run consumes a large volume of environment transitions—thousands of parallel rollouts advanced over hundreds to thousands of iterations—and the rate at which those transitions are produced sets the wall-clock cost of training. The NRD models run orders of magnitude faster than the simulator they stand in for: batched on a single GPU the conditioned HMMWV model advances at $\approx 1{,}750\times$ real time, the tracked vehicle at $\approx 16{,}300\times$, and the arm at $\approx 2{,}320\times$ (Appendix A). Under the reported aggregate

batched-GPU versus single-process Chrono configurations, the measured simulated-time throughput ratios reach about 11,500× for CRM soil, 58,400× for the tracked vehicle, and 8,300× for the arm; the tracked-vehicle ratio is asymmetric, since Chrono simulates the full vehicle-plus-arm scene while each surrogate models a single operating mode. That margin turns policy optimization from impractical into routine: the tracked vehicle driving policy reaches its plateau in ≈ 10 min and the arm policy trains in ≈ 57 min on a single GPU, each converging within a few hundred PPO iterations (Sec. 5.4).

A single converged run, though, understates what throughput buys. The time-consuming part of applying RL is not that run but the iterative design around it—shaping the reward and tuning the optimizer until the policy behaves as intended, each adjustment validated only by training again. Both policies needed such work: spin and drive-against-brake penalties for the vehicle, and reward shaping and safety-interface design for the arm—the terminal bonus, action regularization, clearance observation, and command shield (Sec. 5.4). The effective cost of a policy is therefore one run times the number of design iterations, so a run that finishes in minutes rather than the orders-of-magnitude greater wall-clock cost implied by the measured single-process Chrono throughput keeps that loop interactive—more reward and hyperparameter iterations per day, which is what ultimately yields a working policy.

6.5 High-Fidelity Validation Is the Final Test

An NRD model is validated at three levels of increasing stringency. One-step prediction is the most basic: given a ground-truth history, it checks whether the model has captured the single-step input-to-output relation, and so tests the soundness and capacity of the learned map. It is necessary but limited, because a low one-step error does not guarantee that repeated application stays on trajectory. Open-loop rollout is the more informative metric: feeding the model its own predictions over a horizon measures how faithfully it reproduces Chrono's behavior as a dynamical system, which is the fidelity that governs downstream control. In both case studies the NRD model used to train RL is therefore chosen by minimum open-loop rollout error rather than by one-step loss (Sec. 4.3, Sec. 5.3). Closed-loop transfer into Chrono is the final level and the exam for the whole pipeline: the trained policy parameters are transferred unchanged, while Chrono reproduces the same observation construction, action processing, and safety interface used during policy training inside the NRD model, testing the NRD model and the policy together.

The third level is decisive, because the first two do not by themselves establish that the abstraction is useful for control. A policy can earn high reward inside an approximate model by exploiting its errors, so success in the NRD model environment is necessary but not sufficient. What settles the question is Chrono: the tracked vehicle reaches 100 of 100 seeded goals within 0.75 m and the arm 97 of 100 within 0.05 m with zero contacts and joint-limit violations (Sec. 5.5), while the terrain-aware HMMWV policy attains low tracking error on flat, CRM, and zero-shot bumpy terrain—matching or improving on each single-terrain specialist even on its own training terrain (Appendix B). In every case the reduced model is judged not by its own reward but by whether the policies it trains remain effective in the full simulator, and that transfer is the ultimate evidence that the abstraction was the right one.

7 Limitations and Open Problems

The present work is validated entirely in simulation. Returning the learned policies to Chrono tests whether each NRD model retains sufficient task-relevant dynamics for control, but this remains sim-to-sim transfer and does not establish performance on physical systems. The NRD models are trained on and evaluated against a single high-fidelity reference, so whatever bias Chrono's tire, terramechanics, and actuator models carry is inherited by the surrogate rather than exposed by it. Both the NRD models and the policies also operate on compact simulator states rather than sensory observations: the reduced state is read directly from the simulator through the projection P (Sec. 3.1), and the derived quantities the policy observes—vehicle pose, end-effector position, and clearance—are computed from that privileged

state, thereby bypassing perception and state-estimation errors. Extending the framework to vision-based dynamics models and policies, in the manner of latent world models learned from observations [12, 13], followed by hardware validation, is an important next step.

The construction of each reduced state also remains partly heuristic. The design rule of Sec. 3.2 states what the reduced state should retain, but applying it—selecting recurrent variables, analytically reconstructed quantities, context inputs, and operating ranges—requires physical insight and task-specific experimentation. Although this process produces interpretable abstractions, it requires substantial manual effort for each new system. In future work we will introduce an automated or agentic abstraction-design procedure that proposes candidate state projections, reconstruction mappings, and context variables, and evaluates them jointly using rollout accuracy, policy performance, transfer fidelity, and computational cost. The validation hierarchy used here (Sec. 6.5) already supplies the selection signal such a search would optimize.

Finally, the framework was not evaluated on explicit contact-mode transitions. Although the vehicle cases include aggregate tire–terrain interaction, the arm model is trained only on free-space motion and its controller avoids contact through the safety shield (Appendix F); grasp acquisition and release, impacts, and object–object collisions therefore remain outside the present scope. These events introduce discontinuous and potentially multimodal dynamics that are difficult to represent reliably with a single continuous learned transition model. Extending the method to contact-rich manipulation will likely require event- or contact-conditioned neural dynamics, in which a contact mode is predicted and the transition is conditioned on it [15, 18], combined with reduced deformable simulation approaches such as Simplicits and FreeForm [20, 38] for the deformable and contact-mediated physics that a compact rigid-state abstraction cannot represent.

8 Conclusion

This paper presented a framework for distilling high-fidelity robot simulations into task-specific neural reduced dynamics models that serve as massively parallel environments for policy learning. The central result is that useful reduction is determined by control relevance rather than dimensionality alone. Each abstraction retains the recurrent physical variables and context needed to propagate task behavior, reconstructs known kinematic quantities analytically outside the network, and is evaluated through sustained open-loop rollouts and, ultimately, closed-loop transfer back to Chrono.

Across the HMMWV, tracked-vehicle, and arm tasks, three distinct abstractions supported policy optimization entirely within frozen, vectorized NRD models. The HMMWV generalist achieved lower median and mean tracking error than both single-terrain specialists on every evaluated terrain, including the zero-shot bumpy case. When transferred to Chrono, the tracked-vehicle and arm policies reached 100 of 100 goals within 0.75 m and 97 of 100 within 0.05 m, respectively, with no arm contacts or joint-limit violations. In both case studies the NRD model advanced its task roughly four orders of magnitude faster in simulated time than the high-fidelity scene it replaced, under the reported batched-GPU surrogate and single-process Chrono configurations: $\approx 11{,}500\times$ for CRM soil in Case Study I, and $\approx 8{,}300$–$58{,}400\times$ across the two control modes of Case Study II. Together, these results show that, within the tested simulator regimes, task-designed NRD models can retain the dynamics required for control while moving high-volume policy optimization out of the expensive high-fidelity simulator. Chrono remains the high-fidelity reference and final validation authority.

Statements and Declarations

Funding

This work was supported in part by National Science Foundation grant CMMI-2153855.

Author contributions

Harry Zhang: Conceptualization, Methodology, Software, Investigation, Validation, Visualization, Writing – original draft. **Dan Negrut**: Conceptualization, Supervision, Funding acquisition, Resources, Writing – review & editing. Both authors read and approved the final manuscript.

Data and code availability

The datasets, trained model checkpoints, experiment configurations, and the source code for data generation, NRD training, policy training, and closed-loop Chrono validation are publicly available on the project page, https://uwsbel.github.io/NeDM/ [36].

Appendix A Simulation Throughput and Real-Time Performance

The neural reduced dynamics model is motivated by the wall-clock cost of the high-fidelity Chrono simulator, which grows sharply with contact-model complexity. We probed the stepping throughput of each Chrono scene used in this paper with the same collector/evaluation scene builders that generate the data (measured on a single Intel 14900KF, one process, throttle 0.4), and separately measured each case study's NRD model throughput as a batched GPU forward pass during its own PPO training run. The cross-case comparable metric is the real-time factor (RTF), i.e., simulated seconds advanced per wall-clock second (RTF = 1 is real time); Cost is normalized to each case's NRD model (1×), so it reads directly as how many times more wall-clock time high-fidelity simulation costs than the surrogate that replaces it. Results are split by case study: Table 9 (Case Study I) and Table 10 (Case Study II).

A.1 Case Study I: Terrain-Aware HMMWV

Flat rigid HMMWV runs at 7.31× real time; a bumpy heightmap costs only ∼9% more (RTF → 6.65×), isolating the heightmap-collision overhead from an otherwise identical vehicle and step size. CRM is the dominant Chrono cost at 0.152× RTF (∼14.1 M SPH particles, ∼10.8 M boundary markers, and a 4× smaller step, not the vehicle model), so a 15 s CRM episode takes ∼100 s of wall-clock time. The 8-layer NRD model sustains ≈ 175,000 dynamics passes/s aggregate (≈ 1,750× RTF) independent of terrain (Sec. 4.3); normalized to it, even flat-rigid Chrono costs 239× more wall-clock time per simulated second, and CRM costs 11,500× more.

Table 9 Case Study I (terrain-aware HMMWV) throughput. Chrono rows share the fixed `HMMWV_Full` configuration and differ only in the tire–terrain contact model. Cost is normalized to the NRD model row (1×)

Configuration	Δt	Steps/s	RTF	Cost
HMMWV — rigid flat	2.0 ms	3,657	7.31×	239×
HMMWV — bumpy rigid	2.0 ms	3,324	6.65×	263×
HMMWV — CRM soil	0.5 ms	304	0.152×	11,500×
HMMWV — NRD model (8-layer)[a]	10 ms	175,000	1,750×	1×

[a] The NRD model row is an aggregate batched-GPU figure, not a single-stream process like the three Chrono rows above: Steps/s is dynamics forward passes/s summed across the 2,048 parallel rollouts used during PPO training (Sec. 4.5), read from the rsl-rl training logs (`Perf/total_fps`, consistently ≈ 35,000 policy-control steps/s across all three 8-layer policies, × action_repeat = 5); RTF is the corresponding aggregate simulated-seconds advanced per wall-clock second.

Table 10 Case Study II (tracked vehicle + arm) throughput, measured the same way as Table 9. The single Chrono vehicle-plus-arm scene serves both control modes; each NRD model row is its own 1× reference, and the Chrono Cost cell lists the wall-clock ratio over the drive-mode / reach-mode NRD model respectively

Configuration	Δt	Steps/s	RTF	Cost
Tracked vehicle + arm — rigid	0.5 ms	558	0.279×	58,400× / 8,300×
Tracked vehicle — NRD model[a]	20 ms	815,000	16,300×	1×
Arm — NRD model[a]	20 ms	116,000	2,320×	1×

[a] As in Table 9, both NRD model rows are aggregate batched-GPU figures (not single-stream like the Chrono row): Steps/s is dynamics forward passes/s during PPO, read from the rsl-rl logs as `Perf/total_fps` × action_repeat. Drive mode uses `tracked_transformer_v1` (2,048 rollouts, `Perf/total_fps` ≈ 163,000 × action_repeat 5); reach mode uses the larger `arm_transformer_8d_v1` (4,096 rollouts, `Perf/total_fps` ≈ 116,000, action_repeat 1). The Chrono scene simulates the full coupled vehicle-plus-arm system, whereas each NRD model represents only the subsystem its mode controls.

A.2 Case Study II: Tracked Vehicle + Arm

The tracked drive-mode NRD model is smaller than the HMMWV backbone (3 layers, 96-D embedding, 20 ms step) but still batched across 2,048 rollouts, sustaining ≈ 815,000 dynamics passes/s (≈ 16,300× RTF); the larger reach-mode arm NRD model (5 layers, 256-D, batched across 4,096 rollouts) sustains ≈ 116,000 passes/s (≈ 2,320×

RTF). The single-process Chrono tracked-vehicle-plus-arm simulation, at 0.279× RTF (reflecting the small integration step and stiff solver that single-pin track contact requires), costs ≈ 58,400× more wall-clock time per simulated second than the drive-mode surrogate and ≈ 8,300× more than the reach-mode surrogate—the former is the largest gap of either case study, though the comparison is not fully symmetric: the Chrono scene simulates the full coupled vehicle-plus-arm system, whereas each NRD model represents only the subsystem its mode controls.

Appendix B Per-Terrain Policy-Transfer Statistics

Table 11 reports the full closed-loop Chrono tracking statistics summarized by Fig. 5: for each terrain and policy, the median, mean, and interquartile range (IQR) of the XY RMSE over the 20 held-out references. All three policies are trained on the 8-layer dynamics backbone (Sec. 4.3, Appendix C) and evaluated at their 1000-iteration checkpoint under the same action clipping, and every cell completes 20/20 rollouts with no early terminations.

Figure 9 shows the closed-loop Chrono XY paths behind these statistics for three representative held-out maneuvers per terrain (sustained turn, multi-steer, sine-steer; 9 evaluations total), for the reference and all three policies.

Appendix C Dynamics-Architecture Sweep

The 8-layer dynamics backbone used throughout this paper (Sec. 4.3) is the result of a one-factor-at-a-time (OFAT) architecture sweep over four axes: depth ($n_{\text{layer}} \in \{2, 4, 6, 8, 12\}$), width ($n_{\text{embd}} \in \{128, 192, 256, 384, 512\}$, with the attention head count co-scaled so that head_dim $= n_{\text{embd}}/n_{\text{head}}$ stays fixed at 32), attention-head width (head_dim $\in \{16, 32, 64\}$ at fixed $n_{\text{embd}} = 256$), and train-time context (block_size $\in \{32, 64, 128, 256\}$). Each arm varies exactly one axis from a shared 6-layer, 256-embedding, head_dim-32, 128-context configuration (Ablation model 10 in Table 12), holding the training recipe of Sec. 4.3 fixed otherwise; that shared configuration is trained once rather than once per axis, so the four axes contribute 4 (depth) + 4 (width) + 2 (heads) + 3 (context) new configurations to the 1 shared one, 14 in total. Each is trained for the full 80 epochs and ranked by the same domain-balanced selection score $S = \frac{1}{2}E_{\text{rigid}} + \frac{1}{2}E_{\text{CRM}}$ of Eq. (9) used to pick every model's checkpoint.

Appendix D Training-Data Scaling

Holding the 8-layer backbone (Appendix C) and the rest of the training recipe (Sec. 4.3) fixed, we retrain it on nested seeded subsets of 20%, 40%, 60%, and 80% of the training episodes, applied to both the flat and CRM sources so the 75/25 mix is preserved; compute is held fixed at 80 epochs × 2000 mini-batches by sampling with replacement over the smaller pool, so unique-trajectory count is the only variable. The 100% point is the same run used throughout the paper (Ablation model 1 of Table 12).

S falls from 6.9% to 4.3% over the 20–80% range—more training data consistently buys rollout accuracy. The 100% point does not continue that trend in this single training seed ($S = 4.6\%$, marginally above the 80% point), which we read as seed noise rather than a genuine data-quantity regression, since nothing else changes between the two runs and every other step of the 20–80% curve is monotonic. We use the full dataset as the baseline reported throughout the paper: the scaling trend gives no reason to expect held-back data to hurt, and 100% removes the arbitrariness of choosing a fixed held-out fraction of otherwise-usable training trajectories.

Appendix E Reduced-State and Context Ablation

Sec. 4.2 and Sec. 4.3 motivate the 15-D reduced state and the one-hot terrain code from first principles—the four tire normal forces and four wheel speeds are argued to be what makes slip and load transfer observable, and the terrain key is argued to be what lets one shared backbone disambiguate regimes. Here we test both claims directly by removing each block from the 8-layer backbone (Appendix C) and retraining, holding the rest of the recipe fixed (8 layers, 8 attention heads, 256-D embedding, 128-step context,

Table 11 Closed-loop Chrono tracking statistics (XY RMSE, meters) for the mixture generalist and the rigid-only / CRM-only specialists, on held-out rigid flat, CRM, and zero-shot bumpy references; these are the numbers summarized by Fig. 5. Median, mean, and IQR are over 20 references at the 1000-iteration checkpoint, all 20/20 successful. Bold marks the best median and mean in each terrain block; on this 8-layer backbone the mixture generalist takes every bold cell, i.e. it achieves the lowest mean and median XY RMSE on all three evaluated terrains

Terrain	Policy	Median	Mean	IQR
Rigid flat	Mixture generalist	**0.157**	**0.184**	0.099–0.236
	Rigid-only	0.174	0.219	0.163–0.277
	CRM-only	0.232	0.259	0.189–0.328
CRM	Mixture generalist	**0.180**	**0.249**	0.134–0.321
	Rigid-only	0.854	1.000	0.365–1.447
	CRM-only	0.231	0.361	0.174–0.521
Rigid bumpy[a]	Mixture generalist	**0.149**	**0.229**	0.099–0.247
	Rigid-only	0.187	0.238	0.159–0.290
	CRM-only	0.213	0.418	0.166–0.416

[a] Bumpy is a strict zero-shot test: rigid code, no bumpy data in training or selection.

Table 12 All 14 configurations in the dynamics-architecture sweep, sorted by selection score S (Eq. 9; lower is better), each at its own best epoch. Each model varies exactly one of depth (n_{layer}), width (n_{embd}, with the attention head count co-scaled so head_dim $= n_{\text{embd}}/n_{\text{head}}$ stays fixed at 32), attention-head width (head_dim, at fixed $n_{\text{embd}} = 256$—the axis that actually changes head_dim), or train-time context (ctx) from the shared 6-layer/256-embedding/head_dim-32/128-context configuration (Ablation model 10). *One-step loss* is the channel-reweighted Huber validation loss as training (Eq. 8); *rollout err/dist* is the 10 s open-loop position error normalized by distance traveled. The top row (bold), Ablation model 1, is the 8-layer configuration used throughout this paper

Model	n_{layer}	n_{embd}	head_dim	ctx	One-step loss		10 s rollout err/dist		S
					Flat	CRM	Flat	CRM	
Ablation model 1 (used)	**8**	**256**	**32**	**128**	**0.0025**	**0.094**	**3.7%**	**5.4%**	**4.6%**
Ablation model 2	6	256	32	256	0.0021	0.114	6.1%	3.7%	4.9%
Ablation model 3	6	512	32	128	0.0020	0.103	4.6%	5.3%	4.9%
Ablation model 4	6	384	32	128	0.0030	0.095	5.6%	4.9%	5.2%
Ablation model 5	6	256	16	128	0.0021	0.097	6.7%	3.9%	5.3%
Ablation model 6	12	256	32	128	0.0034	0.091	6.4%	4.8%	5.6%
Ablation model 7	6	192	32	128	0.0024	0.087	5.8%	6.1%	5.9%
Ablation model 8	2	256	32	128	0.0040	0.097	9.1%	4.4%	6.8%
Ablation model 9	6	256	64	128	0.0021	0.095	7.5%	6.1%	6.8%
Ablation model 10	6	256	32	128	0.0022	0.095	9.1%	5.8%	7.4%
Ablation model 11	6	256	32	64	0.0023	0.090	6.9%	8.3%	7.6%
Ablation model 12	4	256	32	128	0.0034	0.091	8.7%	7.0%	7.8%
Ablation model 13	6	256	32	32	0.0035	0.101	9.4%	6.9%	8.2%
Ablation model 14	6	128	32	128	0.0037	0.092	8.8%	7.7%	8.3%

Table 13 Data-quantity ablation at the 8-layer backbone: selection score S (Eq. 9) and its one-step-loss/rollout-err components at each run's own best epoch, as a function of the fraction of training episodes used

Data	One-step loss		10 s rollout err/dist		S
	Flat	CRM	Flat	CRM	
20%	0.0023	0.138	4.2%	9.6%	6.9%
40%	0.0022	0.124	8.1%	4.8%	6.5%
60%	0.0021	0.114	7.0%	5.7%	6.3%
80%[a]	0.0020	0.105	3.9%	4.6%	4.3%
100% (used)	**0.0025**	**0.094**	**3.7%**	**5.4%**	**4.6%**

[a] 80% attains a marginally lower single-seed S than the full-data point; see the discussion below.

75/25 flat/CRM mix, equal-domain-combined-std Huber, `rollout_sel` checkpoint selection, AdamW 3e−4 → 3e−5, 80 × 2000 steps, single seed). The two tests probe different parts of the design: the first removes the terrain-regime context supplied to the model, the second changes the reduced-state

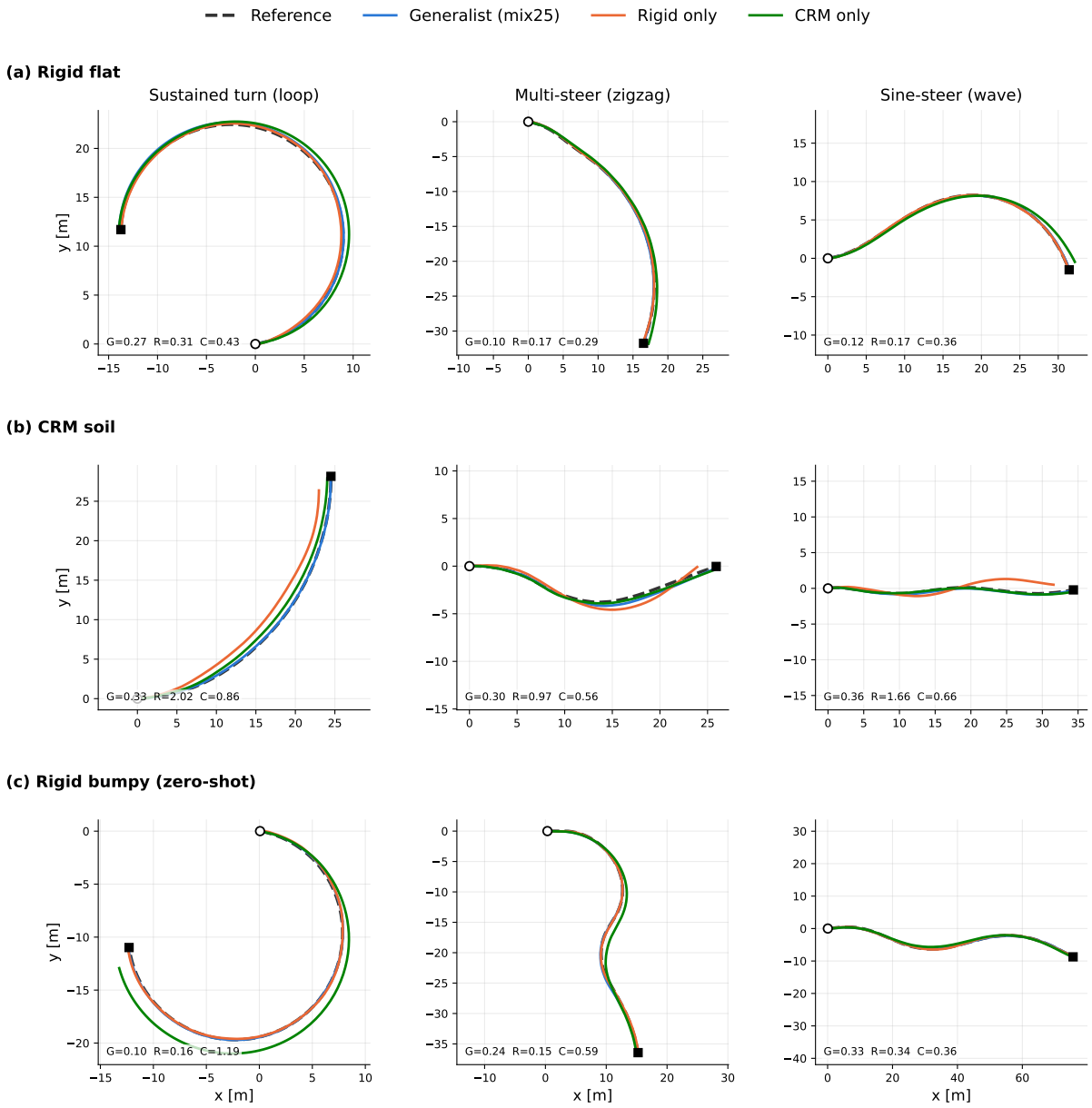


Fig. 9 Closed-loop XY paths for three held-out maneuvers (columns) on each terrain (rows): (a) rigid flat, (b) CRM soil, (c) rigid bumpy (zero-shot). Each panel is annotated with the episode's XY RMSE (m) for the generalist (G), rigid-only (R), and CRM-only (C) policies. The rigid-only specialist (never trained on CRM) visibly overshoots every turn in row (b), consistent with its 1.000 m mean RMSE on CRM in Table 11; the CRM-only specialist is the outlier in row (c) despite bumpy sharing its rigid contact code. Full 20-reference statistics for every maneuver are in Table 11

definition itself—both the recurrent state and the prediction target.

- **Context ablation** (`no_onehot`): drops the 2-D terrain key, input 20-D → 18-D, readout unchanged at 15; the model must infer rigid-vs-CRM from the state history alone.
- **State-abstraction ablation** (`no_tireforce_omega`): drops the four tire normal forces $F_z^{(i)}$ and four wheel speeds $\omega^{(i)}$ from both the recurrent state and the prediction target, input 20-D → 12-D, readout 15 → 7 (body kinematics only: $v_x, v_y, \phi, \theta, \omega_x, \omega_y, \omega_z$). This is

Table 14 Reduced-state and context ablation at the 8-layer backbone, metrics at each run's own best-validation checkpoint. One-step loss is the channel-reweighted Huber validation loss; 10 s open-loop err/dist is the open-loop rollout position error normalized by distance traveled, 12 episodes/domain

Model	One-step loss		10 s open-loop err/dist	
	Flat	CRM	Flat	CRM
Baseline (20-D)	**0.00248**	**0.09378**	**0.0373**	**0.0538**
No one-hot (18-D)	0.00236	0.09395	0.0826	0.0486
No terramechanics (12-D)	0.00341	0.06198[a]	0.0341	0.0800

[a] Computed over 7 readout channels instead of 15 and not comparable to the other two rows (see text); the open-loop column is comparable across all three, since it is integrated from v_x, v_y, ω_z, which every variant retains.

an ablation of the reduced-state definition, not merely of the model's inputs.

Removing the one-hot barely changes one-step loss (flat 0.00248 → 0.00236, CRM 0.09378 → 0.09395) but more than doubles open-loop flat error (0.0373 → 0.0826) while leaving CRM unchanged. The terrain key is thus largely invisible to one-step loss but important over long rollouts: without the explicit regime label, the shared model does not reliably disambiguate the rigid and CRM transition dynamics from state history alone, producing substantially larger accumulated error on the rigid domain. Removing the terramechanics block is the mirror image: flat open-loop error is unaffected (0.0373 → 0.0341) but CRM rises ∼49% (0.0538 → 0.0800), consistent with Eq. (7), since wheel speed and body velocity are nearly redundant on rigid terrain but decouple under CRM slip. That variant's one-step CRM loss (0.062) is computed over 7 channels instead of 15 and is not comparable to the other rows, so we rank all three on the shared open-loop metric instead. Both ablations are offline NRD-rollout comparisons excluded from the architecture ranking of Table 12, and the 7-D checkpoint is not compatible with the 15-D state consumed by the RL environment (Sec. 4.5), so we do not report a closed-loop transfer comparison for it.

Appendix F Forward-Kinematics Safety Shield for Arm Reaching

The arm NRD model is trained only on free-space motion—transitions in contact with the ground, the vehicle, or the arm itself are excluded from its dataset (Sec. 5.2)—so it has no notion of contact and is unreliable for any configuration that would collide. During policy optimization, where undirected exploration would otherwise drive the arm into such configurations, we keep every commanded configuration inside the free-space envelope with a geometric safety shield, and we expose the arm's proximity to collision to the policy as the clearance channel $\hat{\kappa}$ of the observation in Eq. (17). Both are computed by a batched forward-kinematics (FK) model that runs on the GPU alongside the reduced dynamics; neither invokes Chrono. This appendix details that FK clearance; the per-step rule by which the shield accepts or rejects a proposed command increment using it is given in Sec. 5.4.

F.1 Forward Kinematics and Collision Geometry

The 4-DOF arm is a CAD import with non-DH joint frames, so its kinematics are extracted once from the Chrono scene and reproduced as a product-of-exponentials FK evaluated batched over all parallel rollouts. Unlike the end-effector-only kinematics that the reward needs, the shield needs *every* link pose: the FK returns the world pose of all $L = 7$ collision links (shoulder, biceps, elbow, wrist, end-effector, and the two fingers). Each link is bounded by an axis-aligned box, sampled at its eight corners and center; the mounted vehicle is represented by a single conservative axis-aligned box (padded by 0.2 m), and the ground is the plane $z = z_0$. Figure 10 shows the four arm-chain link boxes placed by this FK for an example configuration.

F.2 Signed Clearance

For a configuration q, FK places the link sample points $\{x_p(q)\}$ in the world. The signed distance from a point x to an axis-aligned box with center c

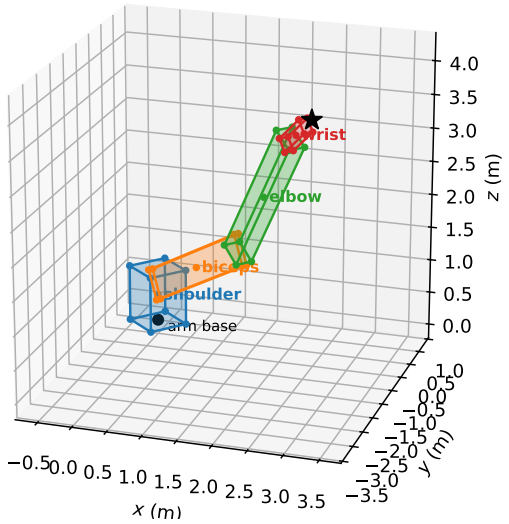


Fig. 10 Forward kinematics placing the arm's collision geometry, for an example configuration $q = [0.70, 0.50, -0.50, 0.30]$ rad. Each of the four arm-chain links (shoulder, biceps, elbow, wrist) carries an axis-aligned box that FK maps into the world as an *oriented* box; the dots are the eight box corners plus center—the sample points $\{x_p(q)\}$ on which the signed-distance test in Eq. (19) is evaluated. The star is the end-effector p^{ee}, and coordinates are in the arm-base frame. The gripper links (end-effector body and two fingers) are omitted for clarity

and half-extents h (expressed in the box frame) is

$$d(x; c, h) = \big\| \, (|x - c| - h)_+ \, \big\|_2 + \min\big(\max_i(|x_i - c_i| - h_i), \; 0\big), \tag{19}$$

where $(\cdot)_+ = \max(\cdot, 0)$ elementwise: the first term is the Euclidean distance to the surface for a point outside the box, and the second is the (negative) penetration depth for a point inside. Because the two branches meet at zero on the box surface, d passes continuously from positive (clear) through zero (touching) to negative (overlapping)—a graded collision signal rather than a binary flag. For each box (c, h) this d is evaluated against every sample point that does *not* belong to that box, batched independently over each arm and each parallel environment. The raw geometric clearance is the minimum of this per-point distance over all $9L$ sample points and the three collision constraints,

$$\kappa(q) = \min\big(\kappa_{\text{ground}}, \; \kappa_{\text{veh}}, \; \kappa_{\text{self}}\big), \tag{20}$$

$$\begin{aligned} \kappa_{\text{ground}} &= \min_{p} \; \big(z_p(q) - z_0\big), \\ \kappa_{\text{veh}} &= \min_{p \in \mathcal{D}} \; d\big(x_p(q); \, c_{\text{veh}}, h_{\text{veh}}\big), \\ \kappa_{\text{self}} &= \min_{(i,j) \in \mathcal{N}} \; \min_{p \in i} \; d\big(x_p^{i \to j}(q); \, c_j, h_j\big), \end{aligned} \tag{21}$$

where $x_p(q)$ are the FK-placed sample points and z_p their heights; $\mathcal{D}$ is the set of points on the distal links (elbow, wrist, end-effector, and fingers) that can reach the vehicle box $(c_{\text{veh}}, h_{\text{veh}})$; $\mathcal{N}$ is the set of ordered non-adjacent link pairs—two links joined by a joint overlap at that joint by construction and would always register contact, so these permanently touching (adjacent) pairs are excluded, leaving only pairs that should never meet, ordered so each is tested in both directions—and $x_p^{i \to j}$ denotes link i's point x_p re-expressed in link j's box-aligned frame, so that the box distance d of Eq. (19) applies. Ground clearance is the one term that reduces to a point-to-plane height rather than a box distance. Because each box is sampled only at its corners and center, a positive $\kappa(q)$ provides a conservative collision-screening signal rather than an exact collision certificate—edge- or face-first overlaps could in principle slip between sample points—while $\kappa(q) < 0$ flags predicted penetration; the zero-contact Chrono evaluation of Sec. 5.5 is the empirical validation that the screen holds in deployment.

F.3 Clearance Observation

The policy observes the clipped, normalized clearance

$$\hat{\kappa} = \text{clip}\big(\kappa(q), \; -\kappa_{\text{c}}, \; \kappa_{\text{c}}\big) \, / \, \kappa_{\text{s}}, \tag{22}$$

with clip level $\kappa_{\text{c}} = 1.0\,\text{m}$ and scale $\kappa_{\text{s}} = 0.5\,\text{m}$. This is effectively an FK-derived proximity sensor: it tells the policy how close the current configuration is to contact and saturates once the arm is comfortably far ($> 1\,\text{m}$) from every obstacle—so the policy can learn collision-aware behavior without the NRD model ever having to represent obstacles.

Concretely, $\hat{\kappa}$ enters through a forward-kinematics side-channel wrapped around the frozen NRD model, evaluated once per control step. Given the current reduced state, the policy proposes an increment Δq^{cmd}; after interpolation, collision screening, and joint-limit clipping, the accepted increment updates the absolute setpoint q^{cmd}, and the dynamics model receives this absolute setpoint as its action and predicts the next reduced state.

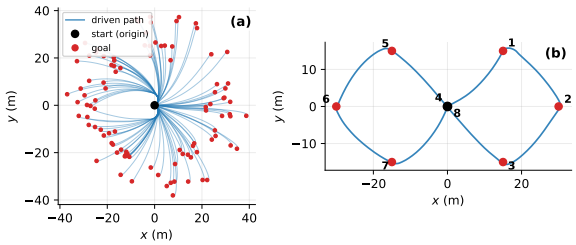


Fig. 11 Tracked-vehicle goal reaching in Chrono under the frozen NRD model policy. (a) All 100 driven paths from the seeded stress battery, each from the common origin (black) to its goal (red). (b) The same policy fed eight consecutive goals along a figure-8 route, each goal numbered by its order in the sequence (goals 4 and 8 return through the start)

Its four joint angles q are read out, FK re-derives the end-effector position and the link boxes from them, and $\kappa(q)$ —normalized to $\hat{\kappa}$ by Eq. (22)—is written into the observation the policy receives on the following step. The NRD model itself therefore never represents the ground, the vehicle, or self-contact: obstacle geometry is injected only through $\hat{\kappa}$, recomputed by FK on the model's own predicted joints at every step.

Appendix G Chrono Policy Evaluation for Tracked Vehicle and Robot Arm

The closed-loop transfer results in Sec. 5.5 come from seeded 100-goal Chrono stress batteries—one per policy—that replace the small spot checks used in early development. Goals are sampled with a fixed seed (12345) from each policy's own trained region, and *each goal is run in its own freshly built vehicle-plus-arm Chrono process*, so no state leaks between goals and the arm's per-goal scene rebuild cannot accumulate; every per-goal trajectory is saved, so success at a tighter tolerance can be recomputed offline without rerunning Chrono. The headline metrics for both batteries are summarized in Table 7; this appendix shows the underlying trajectories.

Tracked vehicle.

Figure 11(a) overlays all 100 driven paths from the stress battery. The policy reaches every goal across the full 360° distribution, including goals behind the vehicle, which it services with forward-loop U-turns rather than in-place spins; the paths are near-direct (median path efficiency 0.959). It stops as soon as it enters the 0.75 m tolerance, so it hugs that radius—median closest approach 0.691 m, with 40/100 landing in the 0.70–0.75 m band—rather than driving all the way onto the goal. Because each goal in the battery starts from the common origin, we additionally chain the *same* policy across a fixed sequence of eight goals, resetting only the target between legs (Fig. 11(b)). Fed the eight vertices of a figure-8 route spanning ±30 m, the point-to-point controller reaches all eight in order (8/8, per-leg closest approach 0.46–0.69 m, 95.5 s total) without any re-planning or per-goal re-tuning—so the single learned goal-reaching skill composes directly into a waypoint-following controller.

Arm.

Figure 12 shows six representative successful reaches chosen for diverse movement—the arm always starts from the same home pose, so the panels differ in reach direction and length (short near reaches, tall vertical extensions, and long lateral and near-horizontal sweeps), each converging the end-effector onto its target within a few centimeters. Across all 100 goals the policy reaches 97 at the 0.05 m tolerance; the three failures are all timeouts (closest approach 6.4–10.9 cm), and no rollout records a contact or joint-limit violation—confirming that the forward-kinematics safety shield (Appendix F) carries from training into high-fidelity Chrono.

References

[1] Mazhar, H., Heyn, T., Pazouki, A., Melanz, D., Seidl, A., Bartholomew, A., Tasora, A., Negrut, D.: Chrono: a parallel multi-physics library for rigid-body, flexible-body, and fluid dynamics. Mechanical Sciences **4**(1), 49–64 (2013) https://doi.org/10.5194/ms-4-49-2013

[2] Tasora, A., Serban, R., Mazhar, H., Pazouki, A., Melanz, D., Fleischmann, J., Taylor, M., Sugiyama, H., Negrut, D.: Chrono: An open source multi-physics dynamics engine. In: Kozubek, T., Blaheta, R., Šístek, J., Rozložník, M., Čermák, M. (eds.) High Performance Computing in Science and Engineering, pp. 19–49. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40361-8_2

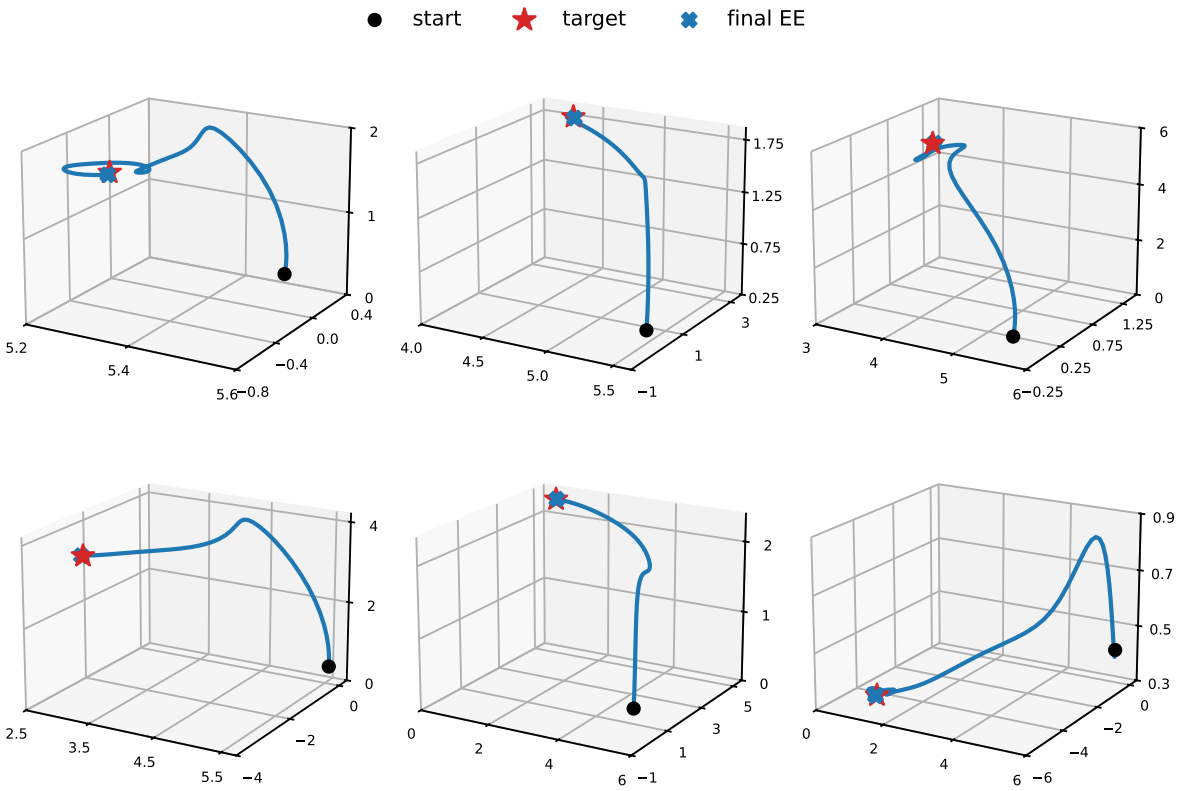


Fig. 12 Six representative successful arm reaches from the seeded 100-goal Chrono stress battery, chosen for diverse movement direction and reach length (the arm always starts from the same home pose). Each panel is the 3-D end-effector path from the start (circle) to the target (star), with the final end-effector position (×); coordinates are in the arm-base frame (drawn z-up)

[3] Serban, R., Taylor, M., Negrut, D., Tasora, A.: Chrono::Vehicle template-based ground vehicle modeling and simulation. International Journal of Vehicle Performance **5**(1), 18–39 (2019) https://doi.org/10.1504/IJVP.2019.097096

[4] Wang, H., Wang, J., Negrut, D.: FNODE: Flow-matching for data-driven simulation of constrained multibody systems. Computer Methods in Applied Mechanics and Engineering **455**, 118912 (2026) https://doi.org/10.1016/j.cma.2026.118912

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

[6] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)

[7] Hu, W., Rakhsha, M., Yang, L., Kamrin, K., Negrut, D.: Modeling granular material dynamics and its two-way coupling with moving solid bodies using a continuum representation and the SPH method. Computer Methods in Applied Mechanics and Engineering **385**, 114022 (2021) https://doi.org/10.1016/j.cma.2021.114022

[8] Unjhawala, H.M., Bakke, L., Zhang, H., Taylor, M., Arivoli, G., Serban, R., Negrut, D.: A physics-based continuum model for versatile, scalable, and fast terramechanics simulation. Journal of Terramechanics **124**, 101150 (2026) https://doi.org/10.1016/j.jterra.2026.101150

[9] Chua, K., Calandra, R., McAllister, R., Levine, S.: Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 31, pp. 4754–4765 (2018)

[10] Janner, M., Fu, J., Zhang, M., Levine, S.: When to trust your model: Model-based policy optimization. In: Advances in Neural Information Processing Systems (NeurIPS), vol. 32, pp. 12519–12530 (2019)

[11] Ha, D., Schmidhuber, J.: World Models (2018). https://doi.org/10.48550/arXiv.1803.10122 . https://arxiv.org/abs/1803.10122

[12] Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., Davidson, J.: Learning latent dynamics for planning from pixels. In: Proceedings of the 36th International Conference on Machine Learning (ICML), pp. 2555–2565 (2019)

[13] Hafner, D., Lillicrap, T., Ba, J., Norouzi, M.: Dream to control: Learning behaviors by latent imagination. In: International Conference on Learning Representations (ICLR) (2020)

[14] Levy, J., Westenbroek, T., Huang, K., Palafox, F., Yin, P., Omidshafiei, S., Kim, D.-K., Gupta, A., Fridovich-Keil, D.: Simulation distillation: Pretraining world models in simulation for rapid real-world adaptation. arXiv preprint arXiv:2603.15759 (2026). To appear at Robotics: Science and Systems (RSS) 2026

[15] Xu, J., Heiden, E., Akinola, I., Fox, D., Macklin, M., Narang, Y.: Neural robot dynamics. arXiv preprint arXiv:2508.15755 (2025). Published at Conference on Robot Learning (CoRL) 2025

[16] Moore, S.A., Lee, E., Chen, B.: Learning legged mpc with smooth neural surrogates. arXiv preprint arXiv:2601.12169 (2026)

[17] Altawaitan, A., Atanasov, N.: Adapting neural robot dynamics on the fly for predictive control. arXiv preprint arXiv:2604.04039 (2026)

[18] Jing, C., Bandi, J.K., Ye, J., Duan, Y., Abbeel, P., Wang, X., Yi, S.: Contact-aware neural dynamics. arXiv preprint arXiv:2601.12796 (2026)

[19] Benner, P., Gugercin, S., Willcox, K.: A survey of projection-based model reduction methods for parametric dynamical systems. SIAM Review **57**(4), 483–531 (2015) https://doi.org/10.1137/130932715

[20] Xiang, D., Modi, V., Dagli, R., Trusty, T., Daviet, G., Chen, A.H., Sharp, N., Levin, D.I.W.: Freeform: Reduced-order deformable simulation from particle-based skinning eigenmodes. arXiv preprint arXiv:2605.29318 (2026). To appear at CVPR 2026

[21] Romero, C., Casas, D., Pérez, J., Otaduy, M.A.: Learning contact corrections for handle-based subspace dynamics. ACM Transactions on Graphics (Proc. SIGGRAPH) **40**(4) (2021) https://doi.org/10.1145/3450626.3459875

[22] Ly, N., Tatsuoka, C., Nagaraj, J., Levy, J., Palafox, F., Fridovich-Keil, D., Lu, H.: Data-driven modeling and correction of vehicle dynamics. arXiv preprint arXiv:2512.00289 (2025). Also published in Journal of Machine Learning for Modeling and Computing, vol. 7, issue 2, 2026

[23] Otto, S.E., Rowley, C.W.: Linearly recurrent autoencoder networks for learning dynamics. SIAM Journal on Applied Dynamical Systems **18**(1), 558–593 (2019) arXiv:1712.01378

[24] Tasora, A., Mangoni, D., Negrut, D., Serban, R., Jayakumar, P.: Deformable soil with adaptive level of detail for tracked and wheeled vehicles. International Journal of Vehicle Performance **5**(1), 60–76 (2019) https://doi.org/10.1504/IJVP.2019.097098

[25] 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). IEEE

[26] Zakka, K., Tabanpour, B., Liao, Q., Haiderbhai, M., Holt, S., Luo, J.Y., Allshire, A.,

Frey, E., Sreenath, K., Kahrs, L.A., Sferrazza, C., Tassa, Y., Abbeel, P.: MuJoCo Playground: An open-source framework for GPU-accelerated robot learning and sim-to-real transfer (2025). https://arxiv.org/abs/2502.08844

[27] NVIDIA: Isaac Lab: Unified framework for robot learning built on NVIDIA Isaac Sim. Accessed 2026-04-29 (2025). https://github.com/isaac-sim/IsaacLab

[28] Freeman, C.D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., Bachem, O.: Brax: A differentiable physics engine for large scale rigid body simulation. In: Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (2021). https://github.com/google/brax

[29] Genesis Authors: Genesis: A Universal and Generative Physics Engine for Robotics and Beyond. https://github.com/Genesis-Embodied-AI/Genesis. Accessed: [January 5, 2025] (2024)

[30] Lee, H., Kim, T., Mun, J., Lee, W.: Learning terrain-aware kinodynamic model for autonomous off-road rally driving with model predictive path integral control. IEEE Robotics and Automation Letters **8**(11), 7663–7670 (2023) https://doi.org/10.1109/LRA.2023.3318190 arXiv:2305.00676

[31] Gibson, J., Vlahov, B., Spieler, P., Theodorou, E.A.: Multistep belief space dynamics learning for risk-aware control. arXiv preprint arXiv:2605.12628 (2026)

[32] Amine, A., Puri, K.R., Le, V.-A., Mangharam, R.: Nonplanar model predictive control for autonomous vehicles with recursive sparse gaussian process dynamics. In: 2026 IEEE Intelligent Vehicles Symposium (IV) (2026). IEEE

[33] Baxter, J.R., Epureanu, B.I., Jayakumar, P., Ersal, T.: High-speed, all-terrain autonomy: Ensuring safety at the limits of mobility. arXiv preprint arXiv:2603.20525 (2026)

[34] Wu, Z., Song, R., Mundheda, V., Navarro-Serment, L.E., Schoenborn, C., Schneider, J.: Tadpo: Reinforcement learning goes off-road. arXiv preprint arXiv:2603.05995 (2026). Accepted at ICRA 2026

[35] Lee, K., Seo, Y., Lee, S., Lee, H., Shin, J.: Context-aware dynamics model for generalization in model-based reinforcement learning. In: Proceedings of the 37th International Conference on Machine Learning (ICML), vol. 119, pp. 5757–5766 (2020)

[36] Zhang, H., Negrut, D.: NeDM: Neural Reduced Dynamics for Complex Robot Control – Project Page, Code, Dataset, and Videos. https://uwsbel.github.io/NeDM/. Accessed: 2026-08-19 (2026)

[37] Rill, G.: An engineer's guess on tyre parameter made possible with TMeasy. In: Proceedings of the 4th International Tyre Colloquium In. University of Surrey, GB. http://epubs.surrey.ac.uk/807823 (2015)

[38] Modi, V., Sharp, N., Perel, O., Sueda, S., Levin, D.I.W.: Simplicits: Mesh-free, geometry-agnostic elastic simulation. ACM Transactions on Graphics (Proc. SIGGRAPH) **43**(4) (2024) https://doi.org/10.1145/3658184.3658200