AnyCar to Anywhere: Learning Universal Dynamics Model for Agile and Adaptive Mobility

Wenli Xiao*, Haoru Xue*, Tony Tao, Dvij Kalaria, John M. Dolan, and Guanya Shi

***Abstract*— Recent works in the robot learning community have successfully introduced generalist models capable of controlling various robot embodiments across a wide range of tasks, such as navigation and locomotion. However, achieving agile control, which pushes the limits of robotic performance, still relies on specialist models that require extensive parameter tuning. To leverage generalist-model adaptability and flexibility while achieving specialist-level agility, we propose AnyCar, a transformer-based generalist dynamics model designed for agile control of various wheeled robots. To collect training data, we unify multiple simulators and leverage different physics backends to simulate vehicles with diverse sizes, scales, and physical properties across various terrains. With robust training and real-world fine-tuning, our model enables precise adaptation to different vehicles, even in the wild and under large state estimation errors. In real-world experiments, AnyCar shows both few-shot and zero-shot generalization across a wide range of vehicles and environments, where our model, combined with a sampling-based MPC, outperforms specialist models by up to 54%. These results represent a key step toward building a foundation model for agile wheeled robot control. We will also open-source our framework to support further research.**

I. INTRODUCTION

The use of the transformer [5] architecture in contemporary robot learning is ubiquitous across perception [6, 7], planning [8] and control [9] tasks. Reinforcement learning with transformers, such as [10] and [11], is used in many downstream applications in bi-manual manipulation [12], navigation [13], humanoid locomotion [14], and whole-body tele-operation [15, 16]. Vision-language-action (VLA) models such as OpenVLA [17], RT-1 [18], and RT-2 [19] demonstrate the scalability of employing transformers in robotics. These models trained on internet-scale data can generalize knowledge and skills to different complex tasks.

Recent advances in robot learning have also introduced more "specialist" systems that can perform highly agile locomotion tasks [20–23]. In particular, on wheeled robots, previous works have achieved high-speed autonomous driving on racetracks [24], grass fields [25], loose sand [26, 27], and off-road terrain [28, 29]. However, most of these efforts [28, 29] are optimized for specific car models and environments, requiring extensive system identification and model training [25, 26], which is costly to fine-tune and difficult to transfer to other wheeled platforms.

On the other hand, agile wheeled control for safety-critical applications requires precise dynamics modeling when running at high speed, since small errors can lead to catastrophic failures such as crashes [25, 30]. There are works that attempt to mitigate this issue by applying neural system identification [9, 30, 31] to adapt the model to different environmental factors such as tire degradation, ground surface imperfections [31], and towed objects [30]. Nevertheless, these methods still require assumptions about the specific car setup (e.g., size and wheelbase of the car).

A key question that arises is: *Can we train a generalist wheeled-robot dynamics model that achieves the performance of a specialist model for each setup?* In this work, we propose AnyCar (depicted in Figure 2), an initial effort to train a vehicle dynamics transformer that can predict the trajectory of various cars in various settings through in-context adaptation. Our contributions are three-fold:

* These authors contributed equally to this work.
The authors are with the Robotics Institute, Carnegie Mellon University, USA. {wxiao2, haorux, longtao, dkalaria, jdolan, guanyas}@andrew.cmu.edu.
Paper website: https://lecar-lab.github.io/anycar/

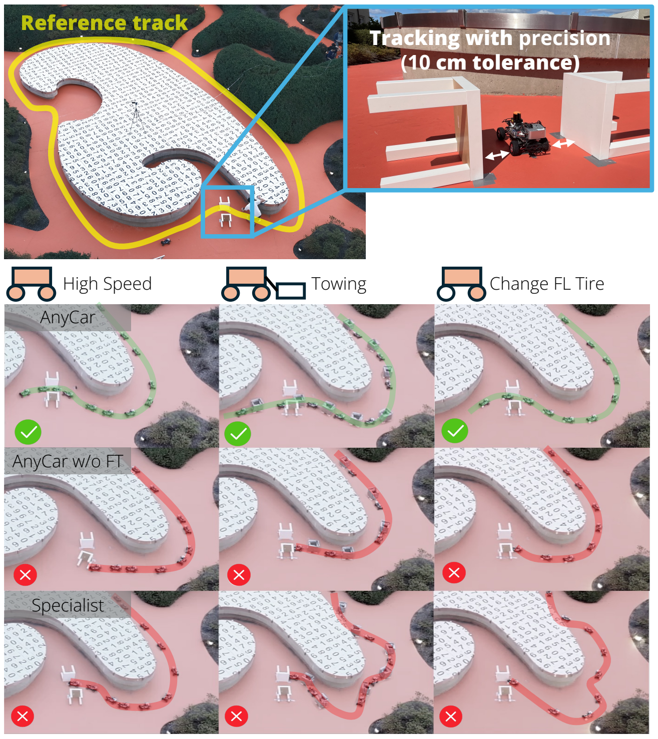

Fig. 1: Performance of AnyCar and baselines in the wild under state estimation errors. Above: A 10 cm tolerance corridor is set as a checkpoint. Below: each row represents the true trajectory of one method, and each column corresponds to a specific setting for the 1/16 scale car: high speed (2 m/s), towing a box, and replacing the front left tire with a plastic wheel. All settings significantly alter the vehicle dynamics.

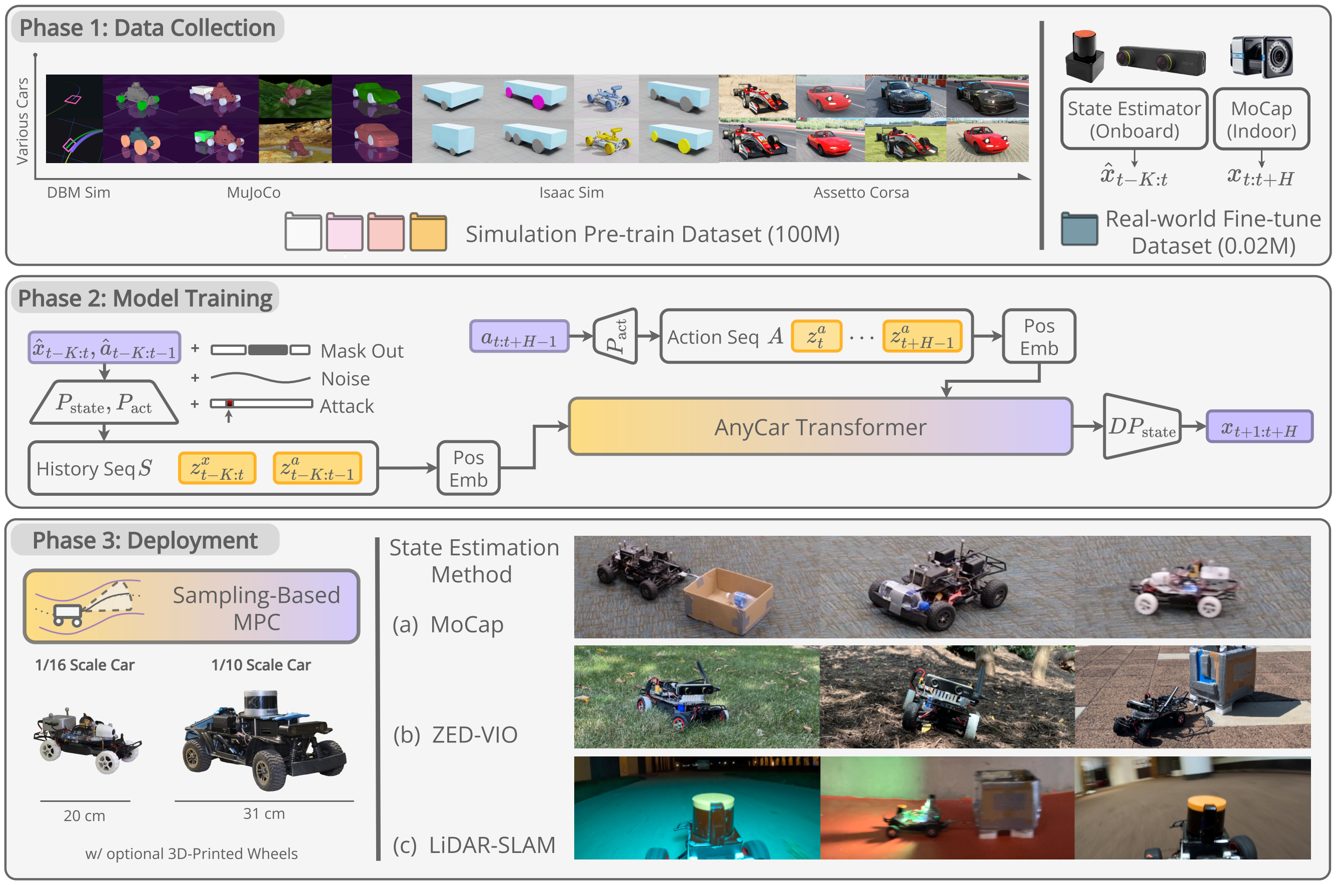

Fig. 2: AnyCar System Pipeline. **Phase 1**: We collect 100M data in 4 different simulations for pre-training and 0.02M few-shot real-world data for fine-tuning the model. **Phase 2**: We pre-train the model with the simulation dataset and enhance prediction robustness through masking, adding noise, and attacking the inputs. We also fine-tune using the fine-tuning dataset. **Phase 3**: We deploy AnyCar in the wild under state estimation error (using SLAM [1–3] and VIO [4]) to control different vehicles (1/10 scale, 1/16 scale) with different settings (tow object, 3D-printed wheels) on different terrains.

- We build a universal synthetic data generator that collects data across diverse vehicles and environments, using physics engines with varying levels of fidelity (e.g., DBM, MuJoCo, Isaac Sim, Assetto Corsa Gym).
- We propose a two-phase robust vehicle dynamics transformer training method that leverages simulation pre-training and real-world fine-tuning to handle sim2real mismatches and state-estimation errors.
- We integrate the dynamics transformer with a sampling-based MPC and demonstrate real-world performance on different car platforms and in different environments, both indoor and outdoor, achieving up to 54% performance improvement over the baseline methods.

II. Related Work

A. Neural Dynamics Model and Adaptive Control

Neural networks, especially transformers, can be used to learn the dynamics of any arbitrary systems [32] or a residual on top of a nominal model, which is expressed in traditional state-space equations [9, 33]. Data-driven techniques can help robots adapt to time-varying dynamics. In particular, open-loop adaptation based on teacher-student training can effectively bridge the Sim2Real gap in RL, such as rapid motor adaptation (RMA) [34]. When real-world ground truth is available, the adaptation can be learned in a more supervised fashion. Safe deep policy adaptation (SafeDPA) [30] performs self-supervised real-world fine-tuning. Neural-Fly [33] trains a residual dynamics model to learn a good representation of environment disturbances with a small amount of real-world data. Learning model predictive control (LMPC) [24, 35] directly regresses a local linear approximation of the state-space dynamics based on a neighborhood of nearest historical states collected in the real world. However, adaptation-at-scale remains an open challenge, particularly when dynamics differ significantly, such as with robots varying in parameters or embodiments.

B. Cross Embodiment

Recent research shows that deep learning models trained at scale can control a variety of robots using the same policy [36, 37]. CrossFormer [36] highlights that training a single transformer on tasks across six embodiments (wheeled robot, quadruped, manipulator, etc.) works even without aligning the action space of these embodiments. The open X-Embodiment dataset [38, 39] focuses on robot manipulation across different embodiments. However, no existing generalist performs agile control. To reconcile generalist adaptability and specialist agility, AnyCar addresses generalization across the same form of embodiment, say, wheeled robots, which is a scope reduction from cross-embodiment.

C. *Transformer for Low-Level Control*

Transformers have shown potential in low-level control, with attention patterns in Trajectory Transformer effectively capturing properties of Markov decision processes (MDPs) [10]. This raises the question of whether transformers, with their inductive bias on pairwise attention between sequence elements, can achieve efficient training and representation of Markovian dynamics for low-level control. Recent work has begun exploring this area. [16] uses a transformer for humanoid whole-body control, though without addressing cross-embodiment generalization. [9] applies a transformer for online system identification in vehicular robots, using historical states to generate a context vector for a neural dynamics model. Despite these advancements, current literature lacks examples of transformers excelling in both specialized tasks (e.g., agile locomotion or dexterous manipulation) and generalization across different robots or tasks.

III. Overview

Notation. x_t, a_t are state and action at time step t. We use $x_{1:t}$ to denote a sequence $\{x_1, \cdots, x_t\}$. $\hat{x}$ denotes estimation of x (e.g., from VIO) and $\hat{a}$ is a with noise.

To demonstrate the practical application of our method, we focus on trajectory tracking in unstructured environments for a range of vehicles, which can be formulated as follows:

$$\begin{aligned} \underset{a_{0:T}}{\text{maximize}} \quad & \sum_{t=0}^{T} R(x_t, a_t) \\ \text{subject to} \quad & x_{t+1} = f(x_t, a_t, c_t), \quad \forall t = 0, 1, \dots, T \end{aligned}$$

where $R(x_t, a_t)$ is the reward function and $x_{t+1} = f(x_t, a_t, c_t)$ represents the system dynamics with c_t representing all physics characteristics related to car dynamics and the environment conditions such as terrain, payload, .etc. The state is defined by $x_t \triangleq [p^x_t, p^y_t, \psi_t, \dot{p}^x_t, \dot{p}^y_t, \omega]$, which contains position (p^x_t, p^y_t), heading angle ψ_t, linear velocity $(\dot{p}^x_t, \dot{p}^y_t)$, and angular velocity ω. The action $a_t \triangleq [\mathcal{T}, \delta]$ contains throttle $\mathcal{T}$ and steering angle δ. Our model is a seq2seq model that can predict the future state sequence from *imperfect* state and action history sequence and future action sequence:

$$x_{t+1:t+H} \approx f^{\text{AnyCar}}_{\theta}(\underbrace{\hat{x}_{t-K:t}, \hat{a}_{t-K:t-1}}_{\text{noisy state and action history}}, \underbrace{a_{t:t+H-1}}_{\text{future actions}}), \quad (1)$$

where K denotes history length and H denotes prediction horizon (illustrated in Figure 2 Phase 2). We train a transformer to approximate Equation (1) for various cars and terrains via its in-context adaptation capability. Our model not only can learn adaptive dynamics but also a filter that can handle noisy state estimation $\hat{x}$ and action $\hat{a}$. Compared with previous works that assumes one specific car model with limited adaptable parameters [9, 24, 26, 30, 31], AnyCar can adapt to various types of car with or without assumptions about the environment. To learn θ, we design two-stage training pipeline. In the first stage, in Section IV, we generate large scale dataset which contains trajectories of various cars in different terrains using different physics simulations; we then pre-train the AnyCar transformer $f^{\text{AnyCar}}_{\theta_{\text{sim}}}$ (Figure 2 Phase 2) in Section IV-B. In the second stage, we collect few-shot real-world data and fine-tune the transformer $f^{\text{AnyCar}}_{\theta_{\text{fine-tune}}}$ in Section IV-C. After these two stages, the model $f^{\text{AnyCar}}_{\theta_{\text{fine-tune}}}$ can accurately predict future trajectory for different cars even under state-estimation error.

With an accurate dynamics model, we apply the Model Predictive Path Integral (MPPI) to perform trajectory tracking. MPPI is a sampling-based MPC approach that minimizes the cost-to-go for sampled trajectories and selects optimal controls by weighting multiple candidate trajectories based on their performance. We choose MPPI for its ability to leverage parallel computation and its training-free nature, with no need for a reduced-order model. In Section V-A, we describe the detailed system design for trajectory tracking using AnyCar transformer with MPPI to achieve control at 50Hz in the real world. Finally, in Section VI, we showcase the deployment of AnyCar in various real-world scenarios, demonstrating its versatility and robustness across different vehicles and environments.

IV. MODEL AND DATA

In this section, we describe the dataset collection and model training strategies of AnyCar. As shown in Figure 2 Phase 1 and Phase 2, we highlight our data collection in massive simulation and few-shot real-world data, and our robust training and finetuning pipeline.

A. *Pre-Training with Massive Simulated Data*

Scene Generation. To collect a diverse dataset, we leverage the low-cost nature of simulation and generate a large amount of simulated data. Our data generation has three sources of diversity: 1) dynamics, 2) scenario, 3) controller. As illustrated in Figure 2 Phase 1, we synthesize trajectories of various cars and terrains via four distinct simulators: Dynamic Bicycle Model [31] (DBM)-based numeric simulation, MuJoCo, IsaacSim, and Assetto Corsa Gym [40].[1]

Curriculum Model Training. To ensure data distribution coverage, we diversify between on-policy and off-policy data via a two-stage curriculum learning method. In stage one, we collect high-volume off-policy data as warm-up, by building a hybrid controller where we use a pure pursuit controller for steering δ and a PD controller for throttle $\mathcal{T}$, to track randomly synthesized reference tracks. After collecting 200M timesteps that are usable for training a general model for non-agile tasks (in which the target velocity is smaller than the physical limit), we switch to stage two, where we deploy an on-policy NN-MPPI controller (described in Section V-A) to

[1]The settings are summarized in https://lecar-lab.github.io/anycar/

track agile trajectories, and periodically update the network with the collected on-policy trajectory.

B. Robust In-Context Adaptive Dynamics Model

Model Structure. As illustrated in Figure 2 Phase 2, the historical states in Equation (1) are linearly projected into a 64-dimensional latent space with an encoder layer $P_{\text{state}}(x) : \mathbb{R}^6 \rightarrow \mathbb{R}^{64}$. The historical actions pass through a different encoder $P_{\text{act}}(a) : \mathbb{R}^2 \rightarrow \mathbb{R}^{64}$ of similar size. We then interleave and stack them to make a complete history token sequence $S^{2K-1\times 64}$, which is supplied to the transformer as the context. The future actions are tokenized with $P_{\text{act}}(a)$ and stacked as a sequence $A^{H\times 64}$. Both S and A sequences are then summed with two learnable positional encoders. After passing the context S and input A to the transformer decoder, we obtain $S'^{H\times 64}$, which is then down-projected into the state-space with $DP_{\text{state}}(x) : \mathbb{R}^{64} \rightarrow \mathbb{R}^6$. This becomes the final state sequence prediction.

Robust Training. To learn a robust model f_θ for Equation (1) under various disturbances, we propose to add three techniques in pre-training, i.e., mask out, add noise, and attack (Figure 2). We implement *mask out* by applying randomized cross-attention mask to transformer, then *add noise* $\epsilon \sim \mathcal{N}(0, \epsilon^{\max})$. We also add *attack*: unreasonably large or small values to random dimensions in history to simulate state estimation errors in the real-world. As discussed in Section IV, these techniques collaboratively improves robustness for successful real-world deployment.

C. Fine-Tuning with state-estimation error reduction

After pre-training $f_{\theta_{\text{sim}}}^{\text{AnyCar}}$, we fine-tune the model with 10 minutes of real-world data (0.02M) to reduce the sim2real gap and mitigate state-estimation error. To collect real-world data, we control the car via joystick to follow a few random curved trajectories in a motion capture field and collect both the state from on-board state-estimators and the ground truth states provided by the Vicon system, as shown in Figure 2 Phase 1. We then fine-tune $f_{\theta_{\text{sim}}}^{\text{AnyCar}}$ on collected data under the constraint of $||\theta_{\text{fine-tune}} - \theta_{\text{sim}}||_2 \leq \epsilon_{\text{tune}}$ to prevent catastrophic forgetting. In practice, we also apply data rehearsal to assist. The model after fine-tuning $f_{\theta_{\text{fine-tune}}}^{\text{AnyCar}}$ is ready to deploy at zero-shot.

V. SYSTEM DESIGN

In this section, we describe the system that leverages AnyCar to perform various trajectory tracking tasks.

A. MPPI Controller

As introduced in Section III, we choose to evaluate our method with MPPI instead of RL and MPC, because RL would have required optimizing additional policy and value function parameters, which is outside the scope of this work. Likewise, neural MPC [41] would have resulted in a reduced-order approximation of our neural dynamics model that is undesired for a fair evaluation.

We employ a variant of MPPI called Covariance-Optimal MPC (CoVO-MPC) [42], which improves upon vanilla MPPI with adaptive sampling covariance to achieve an optimal convergence rate on the cost function.

1) Trajectory Sampling: Let x_t, a_t be state and action at time t. For given control horizon H, we randomly sample N action sequences $\{a^i_{t:t+H-1}\}^N_i$ in a normal distribution, whose mean and covariance are computed with [42]. We roll out each action sequence with the AnyCar model and compute their cumulative rewards. In order to generate smooth action sequences, we re-parameterize the control sequence $a_{0:T}$ with a set of time-indexed knots represented by $\theta_{0:k}$ [43]. Given query point τ, the control can be evaluated by $a_\tau = \text{spline}(\tau; (\tau_{0:k}, \theta_{0:k}))$.

2) Reward function: To motivate tracking reference waypoints, we use the single-step reward function from [28], which is given by

$$r(x, a, \hat{x}) = w_1||p-\hat{p}||^2 + w_2||\psi-\hat{\psi}|| + w_3||v_x-\hat{v}_x|| + w_4||\delta a||,$$

where p denotes position, ψ is heading angle, v_x is longitude velocity, and δa is action increment.

3) Computation: We implement the MPPI in JAX [44] and the transformer model in TransformerEngine. We evaluate 600 action sequence samples on the model in parallel, achieving 20 ms (50 Hz) real-time performance on a RTX 4090 GPU. We note that further optimizations, such as KV caching, sharing history token attentions between samples, and TensorRT conversion can enable edge deployment.

B. State Estimation

In our experiments with ground truth data, we use a motion capture system to directly observe the position and velocity of the vehicle. For in-the-wild experiments, we estimate the linear and angular velocity of the vehicle by fusing motor odometry and IMU data using an Extended Kalman Filter (EKF) [45]. We then correct for the odometry drifting using 2D LiDAR SLAM [1–3], or 3D SLAM with visual-inertial odometry (VIO). We will demonstrate AnyCar's fine-tuning pipeline can adapt to state estimation error in Section VI.

C. Low-Level Controller

Upon receiving a throttle (acceleration) and steering (angle) command, a low-level module maps the throttle to motor current with a simple linear mapping, and maps the steering angle to servo angle. Adapting to the discrepancies in these rough mappings and the latency in actuator responses is also a goal of our fine-tuning pipeline.

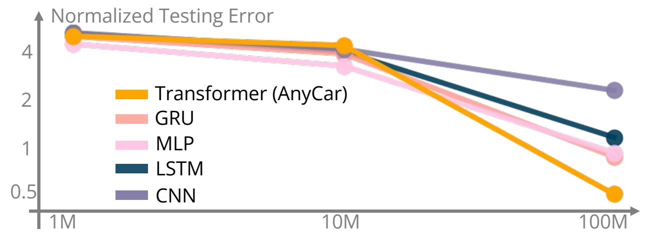

Fig. 3: Comparison of different model structures and data scales. The reported testing error is normalized using the mean and standard deviation of the evaluation dataset.

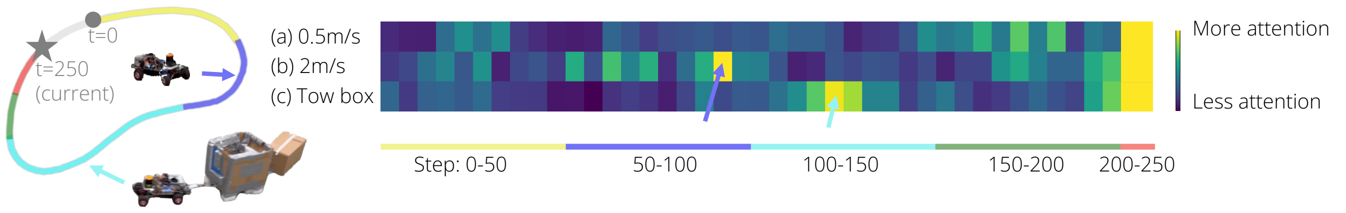

Fig. 4: Visualization of AnyCar's transformer attention in three real-world settings: (a) low speed at 0.5 m/s, (b) high speed at 2 m/s, and (c) towing an object at 2 m/s, all tracking the same reference trajectory. AnyCar's transformer consistently focuses on the nearest 50 steps across all settings and adaptively attends to different sections of the track. For example, it attends to the first corner in setting (b) and the second corner in setting (c).

TABLE I: Indoor Results using Ground Truth State-Estimation (Motion Capture)

Setting	Method	$E^{\text{Prediction}} \downarrow$	$E^{\text{Tracking}} \downarrow$
Few-shot performance in fine-tuned scenarios			
1/10 Scale Car + Tow Box	PP	-	0.45 ± 0.02
	AnyCar (Ours)	0.27 ± 0.24	**0.35** ± 0.04
	AnyCar w/o FT	0.54 ± 0.55	0.47 ± 0.23
	Specialist	**0.14** ± 0.09	0.43 ± 0.05
1/10 Scale Car + Payloads	PP	-	0.54 ± 0.03
	AnyCar (Ours)	**0.11** ± 0.08	**0.41** ± 0.03
	AnyCar w/o FT	0.21 ± 0.11	0.47 ± 0.17
	Specialist	0.26 ± 0.11	0.51 ± 0.03
Zero-shot generalization in unseen scenarios			
1/10 Scale Car + Plastic wheels (All 4 wheels)	PP	-	0.79 ± 0.62
	AnyCar (Ours)	**0.26** ± 0.32	**0.335** ± 0.03
	AnyCar w/o FT	0.32 ± 0.21	0.45 ± 0.15
	Specialist	0.35 ± 0.15	0.334 ± 0.06
1/10 Scale Car + Plastic wheels (Front 2 wheels)	PP	-	0.52 ± 0.05
	AnyCar (Ours)	**0.12** ± 0.08	**0.39** ± 0.05
	AnyCar w/o FT	0.20 ± 0.11	0.49 ± 0.09
	Specialist	0.27 ± 0.11	0.41 ± 0.04
1/10 Scale Car + Plastic wheels + Tow box	PP	-	0.57 ± 0.03
	AnyCar (Ours)	**0.09** ± 0.06	**0.49** ± 0.09
	AnyCar w/o FT	0.18 ± 0.11	0.52 ± 0.09
	Specialist	0.14 ± 0.05	0.60 ± 0.04
1/16 Scale Car + Plastic wheels	PP	-	0.37 ± 0.16
	AnyCar (Ours)	**0.17** ± 0.08	**0.31** ± 0.06
	AnyCar w/o FT	0.25 ± 0.14	0.44 ± 0.08
	Specialist	0.26 ± 0.11	0.55 ± 0.07

VI. Experiment

In this section, we aim to demonstrate the capability of the proposed AnyCar by addressing the following questions:

- **Q1:** Can our model generalize to various cars and terrains, and outperform specialist models?
- **Q2:** Can our model maintain its adaptation capability even with imperfect state estimation?
- **Q3:** Why does the proposed robust vehicle dynamics transformer outperform other baseline models?

Baselines. We compare AnyCar with three baseline methods: 1) *AnyCar w/o FT*: AnyCar without the real-world fine-tuning phase, 2) *PP*: Pure Pursuit controller for steering and PID controller for velocity tracking, and 3) *Specialist*: a DBM model with system identification. We also consider two types of state estimator setups: 1) motion capture (indoor), which can be treated as ground truth, and 2) SLAM [1–3] (indoor or outdoor), which is less accurate than motion capture and much prone to drifting issues.

Metrics. We use the model prediction error $E^{\text{Prediction}} \triangleq ||x^{\text{pred}}_{t+1:t+H} - x^{\text{gt}}_{t+1:t+H}||_2$ to assess prediction accuracy. We also define $E^{\text{Tracking}} \triangleq w_2||p_t - \hat{p}_t||_2 + w_3||v_t - \hat{v}_t||_2$, where w_2 and w_3 are the same weights defined for position and velocity tracking rewards in Section V-A, representing the weighted sum of lateral error and velocity tracking error, to evaluate trajectory tracking performance.

A. Evaluate Model In-context Adaptation Capability

To answer **Q1**, we isolate the estimation error and use motion capture to provide ground truth for state estimation in the real world. An 1/10 Scale Car was employed to tow objects and carry payloads, to create a fine-tuning dataset collected by human teleoperation. This dataset was then used to fine-tune our dynamics model. The model was evaluated across two categories: **few-shot performance** (e.g., towing objects and varying payloads, both present in the fine-tuning dataset) and **zero-shot generalization** (e.g., changing 3D-printed wheels, towing objects with modified wheels, and using a smaller car model with altered wheels). We use a trajectory optimization method [24] to compute an on-the-edge reference trajectory on a raceline. Our method, along with other baselines (AnyCar w/o FT, PP, Specialist), was then deployed to track this trajectory. The evaluation considered both model prediction error and closed-loop tracking error with the full controller. Results in Table I show AnyCar reaching and outperforming baselines in terms of prediction error and tracking error in both few-shot and zero-shot cases. We observed the same "emergent skill" seen in RT-X[38] that after fine-tuning the model with data of one robot (1/10 scale car) performing a certain task (track curves at low speed), a different robot (1/16 scale car) also gets a performance boost in doing a different task (agile raceline tracking).

B. Evaluate Model Capability in the Wild

To address **Q2**, we set up a 2D LiDAR-based SLAM stack on an 1/16 scale car. The car was teleoperated to follow an S-shaped path at low speed, collecting odometry data from both the SLAM system and motion capture to create a small-scale dataset of 24,000 timesteps. The model was then

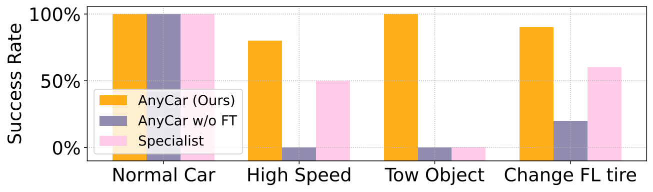

Fig. 5: Comparison with baselines in the wild.

TABLE II: Indoor Results under State-Estimation Error

Setting	Method	$E^{\text{Prediction}} \downarrow$	$E^{\text{Tracking}} \downarrow$
Few-shot performance in fine-tuned scenarios			
1/16 Scale Car + Low Speed	**AnyCar (Ours)**	**0.30** $\pm$ 0.06	**0.35** $\pm$ 0.02
	AnyCar w/o FT	0.32 $\pm$ 0.07	0.48 $\pm$ 0.06
	Specialist	0.33 $\pm$ 0.05	0.42 $\pm$ 0.01
Zero-shot generalization in unseen scenarios			
1/16 Scale Car + High Speed	**AnyCar (Ours)**	**0.44** $\pm$ 0.11	**0.57** $\pm$ 0.03
	AnyCar w/o FT	0.52 $\pm$ 0.15	1.30 $\pm$ 0.11
	Specialist	0.48 $\pm$ 0.10	1.26 $\pm$ 0.89
1/16 Scale Car + Tow 2 Box	**AnyCar (Ours)**	**0.34** $\pm$ 0.10	**0.57** $\pm$ 0.02
	AnyCar w/o FT	0.41 $\pm$ 0.14	1.41 $\pm$ 0.09
	Specialist	0.39 $\pm$ 0.09	0.93 $\pm$ 0.62

fine-tuned using the method in Section IV-C. Using the fine-tuned model (AnyCar), we first evaluated our method in an indoor environment under different conditions (low speed, high speed, towing two boxes) with SLAM, and computed metrics using ground truth data. The results in Table II show AnyCar outperforms baselines with peak improvement of 54%. Based on Table I and Table II, we prove the necessity of fine-tuning that aligning the dynamics model to handle state estimation error.

Next, the system was moved to an outdoor environment without further modification. To validate the robustness to state-estimation capabilities of AnyCar, we set up a narrow corridor along the reference trajectory, allowing the car to pass with a 10 cm tolerance (demonstrated in Figure 1). Poor tracking performance would result in the car colliding with the walls. We compared our method with AnyCar w/o FT and Specialist by calculating the percentage of times the car successfully passed all checkpoints without collision (success rate) for each method. The results in Table II show AnyCar achieves the highest success rate in all settings, with the specialist failing consistently due to the state estimation error. We also visualize the transformer attention across different settings in Figure 4, which demonstrates AnyCar's in-context adaptation in various settings. In addition to SLAM-based state-estimator, we also show the deployment with ZED-VIO [4] on the website.

C. Interpret the expressiveness and robustness of AnyCar

To interpret the success of AnyCar, we argue that scaling and robust training are integral parts of our method. We conducted experiments on datasets of varying sizes, encompassing trajectories of different cars running in different terrains with timesteps ranging from 1 million (1M) to

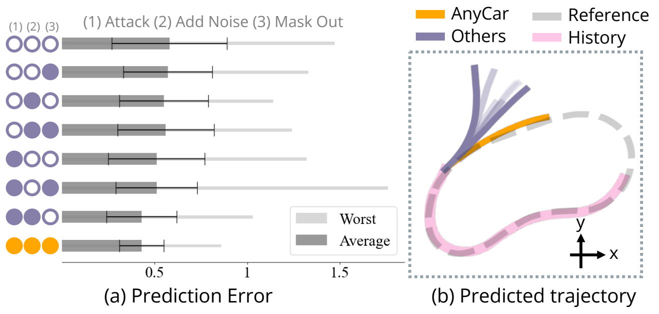

Fig. 6: Comparison of AnyCar robust training methods. (a) Evaluate prediction error in real-world trajectories. (b) Demonstrate predicted trajectory of different methods.

100 million (100M). Various model architectures, including transformer, LSTM, GRU, CNN, and MLP, were evaluated, and the normalized prediction error was calculated for each. To ensure the models could be used with MPPI for 50Hz control, we limited the maximum number of parameters for each model to 200K. We create a testing dataset with 1M data points sampled i.i.d. from the simulation, independent of the training dataset, and evaluate all trained models on the same testing dataset. The results shown in Figure 3 demonstrate that as the training dataset size increased from 10M to 100M timesteps, the prediction errors decreased significantly, with the transformer model performing best at the 100M scale. This highlights that the transformer structure is the most effective for modeling diverse car dynamics and environments compared with baseline models. However, scaling the data and using the appropriate model structure alone are insufficient. Under the optimal data scale and model configuration, we found it crucial to apply the proposed robust training methods (including attack, noise, and mask-out strategies), as discussed in Section IV-B. We systematically evaluate all combinations of these components, resulting in a total of 8 pre-trained models. Each model is fine-tuned using the same dataset and evaluated on real trajectory. The results, shown in Figure 6(a), demonstrate that the model achieves the highest prediction accuracy and stability only when all robust training components are activated. For instance, Figure 6(b) shows that without full robust training, the transformer's predictions are vulnerable to noisy state estimation, leading to significant errors. The combination of model selection, large-scale data, and robust training explains why the AnyCar performs better than the baseline models.

VII. Limitations And Future Work

In this paper, we propose AnyCar, a first step towards foundation model for agile wheeled control. In the future, there are three interesting research directions. One is to use KV caching in the transformer for full onboard computation. The second is to optimize the MPPI control to be aware of model uncertainty and safety. The third is to integrate with existing foundation models for visual navigation [13] to achieve fully agile autonomy in the wild.

ACKNOWLEDGMENT

We express our gratitude to Wennie Tabib, Jiaoyang Li, Jessica Hodgins, and Justin Macey for their assistance with the experiment setup. We also thank Chaoyi Pan and Zeji Yi for the insightful discussions.

REFERENCES

[1] S. Macenski and I. Jambrecic, "SLAM Toolbox: SLAM for the dynamic world," en, *Journal of Open Source Software*, vol. 6, no. 61, p. 2783, May 2021.

[2] S. Macenski, F. Martín, R. White, and J. G. Clavero, "The Marathon 2: A Navigation System," en, in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, arXiv:2003.00368 [cs], Oct. 2020, pp. 2718–2725.

[3] D. Fox, "KLD-Sampling: Adaptive Particle Filters," in *Advances in Neural Information Processing Systems*, vol. 14, MIT Press, 2001.

[4] {Stereolabs}, *Stereolabs/zed-ros2-wrapper*, original-date: 2019-04-12T14:23:46Z, Sep. 2024.

[5] A. Vaswani *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[6] A. Dosovitskiy *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," en, Oct. 2020.

[7] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual Instruction Tuning," en, Nov. 2023.

[8] L. Lehnert *et al.*, "Beyond A*: Better Planning with Transformers via Search Dynamics Bootstrapping," en, Aug. 2024.

[9] S. J. Wang, H. Zhu, and A. M. Johnson, *Pay Attention to How You Drive: Safe and Adaptive Model-Based Reinforcement Learning for Off-Road Driving*, en, arXiv:2310.08674 [cs], Oct. 2023.

[10] M. Janner, Q. Li, and S. Levine, "Offline Reinforcement Learning as One Big Sequence Modeling Problem," in *Advances in Neural Information Processing Systems*, vol. 34, Curran Associates, Inc., 2021, pp. 1273–1286.

[11] L. Chen *et al.*, "Decision transformer: Reinforcement learning via sequence modeling," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, ser. NIPS '21, Red Hook, NY, USA: Curran Associates Inc., Jun. 2024, pp. 15 084–15 097.

[12] T. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware," en, in *Robotics: Science and Systems XIX*, Robotics: Science and Systems Foundation, Jul. 2023.

[13] D. Shah *et al.*, *ViNT: A Foundation Model for Visual Navigation*, en, Jun. 2023.

[14] I. Radosavovic *et al.*, *Humanoid Locomotion as Next Token Prediction*, arXiv:2402.19469 [cs], Feb. 2024.

[15] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang, *Open-TeleVision: Teleoperation with Immersive Active Visual Feedback*, arXiv:2407.01512 [cs], Jul. 2024.

[16] Z. Fu, Q. Zhao, Q. Wu, G. Wetzstein, and C. Finn, "HumanPlus: Humanoid Shadowing and Imitation from Humans," en,

[17] M. J. Kim *et al.*, *OpenVLA: An Open-Source Vision-Language-Action Model*, en, arXiv:2406.09246 [cs], Jun. 2024.

[18] A. Brohan *et al.*, "RT-1: Robotics Transformer for Real-World Control at Scale," en, in *Robotics: Science and Systems XIX*, Robotics: Science and Systems Foundation, Jul. 2023.

[19] B. Zitkovich *et al.*, "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control," en, in *Proceedings of The 7th Conference on Robot Learning*, ISSN: 2640-3498, PMLR, Dec. 2023, pp. 2165–2183.

[20] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi, "Agile But Safe: Learning Collision-Free High-Speed Legged Locomotion," en,

[21] C. Zhang, W. Xiao, T. He, and G. Shi, "WoCoCo: Learning Whole-Body Humanoid Control with Sequential Contacts," en,

[22] T. He *et al.*, "OmniH2O: Universal and Dexterous Human-to-Humanoid Whole-Body Teleoperation and Learning," en,

[23] T. He *et al.*, *Learning Human-to-Humanoid Real-Time Whole-Body Teleoperation*, en, arXiv:2403.04436 [cs, eess], Mar. 2024.

[24] H. Xue, E. L. Zhu, J. M. Dolan, and F. Borrelli, "Learning Model Predictive Control with Error Dynamics Regression for Autonomous Racing," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 13 250–13 256.

[25] K. Stachowicz and S. Levine, *RACER: Epistemic Risk-Sensitive RL Enables Fast Driving with Fewer Crashes*, en, arXiv:2405.04714 [cs], May 2024.

[26] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1433–1440.

[27] G. Williams *et al.*, "Information theoretic MPC for model-based reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1714–1721.

[28] T. Han, A. Liu, A. Li, A. Spitzer, G. Shi, and B. Boots, "Model Predictive Control for Aggressive Driving Over Uneven Terrain," en,

[29] M. Sivaprakasam *et al.*, *TartanDrive 2.0: More Modalities and Better Infrastructure to Further Self-Supervised Learning Research in Off-Road Driving Tasks*, en, Feb. 2024.

[30] W. Xiao, T. He, J. Dolan, and G. Shi, "Safe Deep Policy Adaptation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 17 286–17 292.

[31] D. Kalaria, Q. Lin, and J. M. Dolan, "Adaptive Planning and Control with Time-Varying Tire Models for Autonomous Racing Using Extreme Learning Machine," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 10 443–10 449.

[32] N. Geneva and N. Zabaras, "Transformers for modeling physical systems," *Neural Networks*, vol. 146, pp. 272–289, Feb. 2022.

[33] M. O'Connell *et al.*, "Neural-Fly enables rapid learning for agile flight in strong winds," *Science Robotics*, vol. 7, no. 66, eabm6597, May 2022, Publisher: American Association for the Advancement of Science.

[34] A. Kumar, Z. Fu, D. Pathak, and J. Malik, *RMA: Rapid Motor Adaptation for Legged Robots*, arXiv:2107.04034 [cs], Jul. 2021.

[35] U. Rosolia and F. Borrelli, "Learning Model Predictive Control for Iterative Tasks. A Data-Driven Control Framework," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 1883–1896, Jul. 2018, Conference Name: IEEE Transactions on Automatic Control.

[36] R. Doshi, H. Walke, O. Mees, S. Dasari, and S. Levine, *Scaling Cross-Embodied Learning: One Policy for Manipulation, Navigation, Locomotion and Aviation*, arXiv:2408.11812 [cs], Aug. 2024.

[37] J. Yang *et al.*, *Pushing the Limits of Cross-Embodiment Learning for Manipulation and Navigation*, arXiv:2402.19432 [cs], Feb. 2024.

[38] A. O'Neill *et al.*, "Open X-Embodiment: Robotic Learning Datasets and RT-X Models : Open X-Embodiment Collaboration0," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 6892–6903.

[39] O. M. Team *et al.*, *Octo: An Open-Source Generalist Robot Policy*, arXiv:2405.12213 [cs], May 2024.

[40] A. Remonda *et al.*, *A Simulation Benchmark for Autonomous Racing with Large-Scale Human Data*, arXiv:2407.16680 [cs], Jul. 2024.

[41] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, "Real-Time Neural MPC: Deep Learning Model Predictive Control for Quadrotors and Agile Robotic Platforms," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2397–2404, Apr. 2023, Conference Name: IEEE Robotics and Automation Letters.

[42] Z. Yi, C. Pan, G. He, G. Qu, and G. Shi, *CoVO-MPC: Theoretical Analysis of Sampling-based MPC and Optimal Covariance Design*, arXiv:2401.07369 [cs], Jan. 2024.

[43] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa, *Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo*, en, arXiv:2212.00541 [cs, eess], Dec. 2022.

[44] J. Bradbury *et al.*, *Google/jax*, original-date: 2018-10-25T21:25:02Z, Sep. 2024.

[45] T. Moore and D. Stouch, "A Generalized Extended Kalman Filter Implementation for the Robot Operating System," en, in *Intelligent Autonomous Systems 13*, E. Menegatti, N. Michael, K. Berns, and H. Yamaguchi, Eds., Cham: Springer International Publishing, 2016, pp. 335–348.