Learning to Race Full-Scale Autonomous Racecars

Eric Berndt

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-120
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-120.html

May 16, 2025

Learning to Race Full-Scale Autonomous Racecars

by

Eric Berndt

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Shankar Sastry, Chair
Allen Yang

Spring 2025

Learning to Race Full-Scale Autonomous Racecars

Abstract

Learning to Race Full-Scale Autonomous Racecars

by

Eric Berndt

Master of Science in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Shankar Sastry, Chair

We present a causal transformer, trained via behavior cloning to emulate a model–predictive controller, and demonstrate its zero-shot deployment on a full-scale racecar capable of driving up to 200 mp/h (320 km/h). First, we generate realistic circuits in simulation from online GPS maps, obtaining noisily-measured center-lines and track boundaries that reflect the variability of real venues. Second, we compute time-optimal race lines on every circuit with a spline-based minimum-curvature optimizer. Third, an MPC tracks each race line in simulation while action noise broadens the state distribution; the resulting state–action corpus is tokenized and a transformer is trained to predict future steering angles and drive forces. In simulation, the learned policy closely matches the expert up to 120 mp/h. The model is then deployed zero-shot on the real Las Vegas Road Course, where the network drives the autonomous racecar at 40 mp/h without crashes, with the only degradation being latency-induced oscillations that multi-step prediction alleviates. To our knowledge, this is the first demonstration of transformer-based behavior cloning for a full-scale autonomous racecar and provides the first step towards achieving a superhuman level racer.

Learning to Race Full-Scale Autonomous Racecars

by Eric Berndt

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

[illegible]

Professor Shankar Sastry
Research Advisor

5-16-2025
(Date)

* * * * * * *

[illegible]

Allen Yang
Second Reader

5-16-2025
(Date)

To my father

Who gave me the opportunity he could not have

To my mother

Who championed me when I could not

Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank the mentors and advisors of this project who made this work possible. Shankar Sastry, my advisor, whose leadership formed the foundation of this team. Allen Yang for guiding me with his technical expertise during the formative stages of this project and for his steady guidance within our research team. Gary Passon who made track-side testing days possible through his experience in motor sports and racing operations.

I would like to thank my collaborators on this project who, without their help, this work would not be possible. C.K. Wolfe for her contributions to machine learning and knowledge in simulation. C.K., without you, this race car team would not exist, and your leadership has created an environment for all of us to learn and benefit from. Truly, without your technical contributions, dedication to building team culture, and perseverance through tremendous setbacks, none of this would have been possible. Adith Sundram for his contributions to behavior cloning and for his expertise in controls. Your persistence and friendship throughout the toughest days and challenges kept me on course. Ilija Radosavovic for his guidance on the theory and practical problems in learning-based robotics. Your mentorship formed the basis of this work, and your guidance prevented us from succumbing to many potential missteps.

I would like to thank the AIRacingTech team who worked day and night to make the autonomous race car a reality. Siddarath Saha for the work that built the basis of the race car software. Moises Lopez for your consistent dedication to keeping the race car stable. Edward Lee who went out of his way to provide support. Kevin Chow whose contributions spring boarded me into the team and whose friendship has kept me sane.

I would also like to acknowledge the Indy Autonomous Challenge (IAC) organization, especially Andy Keats.

I would also like to acknowledge the community of fellow course staff members and students of EECS/ME c106A who helped me enter the world of robotics and continue to build the future generations. Thank you to Tarun Amarnath, whose teachings shaped the basis of many future roboticists and taught me how to be a better teacher myself. Your positive attitude shone through even on some of the toughest days our lab has faced.

Chapter 1

Introduction

1.1 Learning in Robotics

The trajectory of robotics research has been significantly reshaped by concurrent breakthroughs in large-scale machine learning. The success of large language models (LLMs) in demonstrating few-shot learning capabilities from vast textual datasets [7] has particularly catalyzed efforts to imbue robotic systems with comparable levels of general-purpose intelligence and adaptability for operation in complex, unstructured environments. This paradigm shift is fostering a new generation of autonomous systems.

A prominent vector in this evolution is the development of embodied foundation models that integrate diverse modalities such as vision, language, and action. These models aim to transfer broad knowledge, often captured from web-scale data, into actionable robotic control. Seminal examples include architectures like PaLM-E, which grounds LLMs in sensorimotor experience [11], and the robotics transformer series (RT-1, RT-2), which have advanced real-world robotic control at scale by learning from extensive, multimodal datasets [6, 5]. More recent innovations such as π_0 [4] and $\pi_{0.5}$ [19] further this trend, focusing on vision-language-action for general robot control and robust open-world generalization, thereby expanding the repertoire of tasks robots can perform with enhanced adaptability.

Simultaneously, reinforcement learning (RL) remains a vital pillar of modern robotics. Foundational algorithms like Proximal Policy Optimization (PPO) [29] and Soft Actor-Critic (SAC) [15] continue to empower robots to acquire intricate skills through direct interaction and feedback. Bridging the persistent sim-to-real gap is a critical challenge, actively addressed by techniques such as domain randomization, which enhances policy robustness by exposing the learning agent to a wide array of simulated environmental variations [31].

These collective advancements are enabling learning-based robotic systems to achieve performance levels that approach or, in specific demanding tasks, even surpass human capabilities.

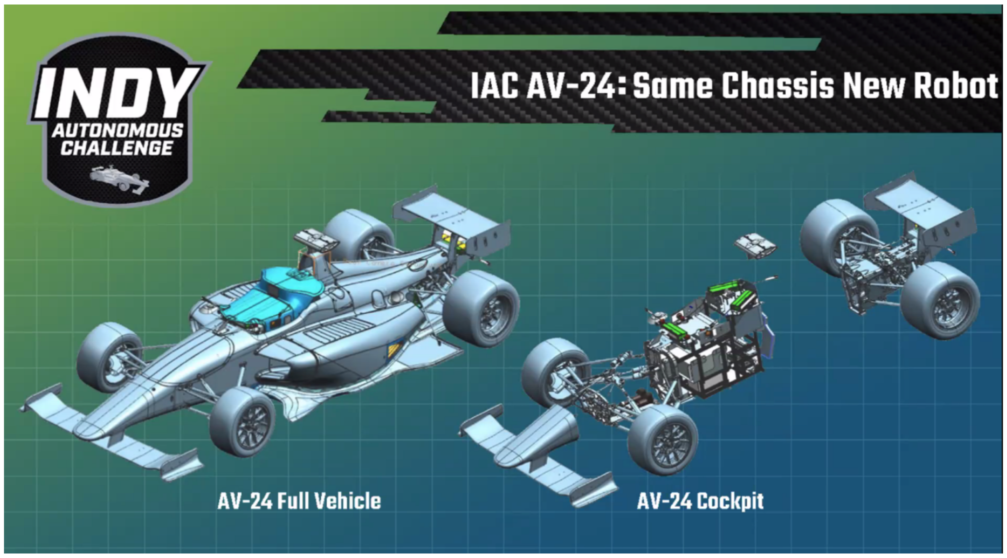


Figure 1.1: AV-24 Chassis

For instance, deep RL has propelled autonomous drones to outperform human world champions in high-speed racing [13] and has allowed autonomous systems to master high-speed flight in unpredictable natural environments [1]. In the realm of legged locomotion, sophisticated learning strategies have endowed quadrupedal [21] and humanoid robots [18, 17, 27] with remarkable agility and resilience, enabling them to navigate challenging, unstructured terrains. These achievements underscore the transformative impact of data-driven learning methodologies in advancing robotic frontiers and provide a compelling context for the research herein, which applies such techniques to the distinct challenges of high-speed autonomous racing.

1.2 Indy Autonomous Challenge

A unique example of extreme robotics performance is the Indy Autonomous Challenge (IAC). The IAC is a global competition in which university teams develop fully autonomous racecars to compete at speeds exceeding 270 km/h (170 mph) on professional racetracks. Unlike traditional autonomous driving benchmarks focused on urban or highway scenarios, the IAC presents a fundamentally different problem: high-speed racing with aggressive maneuvers, sub-second decision-making, multi-agent adversarial interaction, and zero human intervention. The vehicles, called AV-24s, are modified Dallara IL-15 chassis equipped with a standardized suite of sensors and compute, creating a level playing field that emphasizes software innovations over hardware customization.

This multi-agent adversarial component is a crucial differentiator and significantly elevates

Figure 1.2: Team Photo at January CES 2025 - 1st place in the adversarial autonomous head-to-head competition. C.K. Wolfe Team Manager and Simulation Lead, Adith Sundram Controls Lead and Eric Berndt Machine Learning Lead pictured center right

the challenge's complexity. The competition extends beyond single-vehicle time trials to encompass direct, high-speed on-track engagements with other autonomous opponents. These interactions have evolved from structured head-to-head overtaking scenarios, where vehicles must autonomously execute roles as both a lead (defending) car and a following (attacking) car, attempting passes at progressively increasing speeds, to full multi-car races. In these events, several autonomous vehicles compete simultaneously, necessitating real-time navigation of complex traffic situations and dynamic race strategy execution [3]. Such scenarios demand sophisticated learning capabilities far exceeding basic path planning and control. Vehicles must robustly perceive and predict the behavior of other fast-moving autonomous agents, dynamically plan and execute intricate overtaking or defensive maneuvers while adhering to stringent racing rules, and make high-level strategic decisions regarding pace and positioning. All entirely autonomously and often within fractions of a second. This emphasis on direct, high-speed adversarial engagement within a standardized hardware platform establishes the IAC as a critical proving ground for advanced machine learning in physical, safety-critical systems.

AIRacingTech is an IAC university team led by the University of California, Berkeley in collaboration with the University of California, San Diego; Carnegie Mellon University; and the University of Hawai'i, Maui.

1.3 AV-24 Specifications

The technical specifications of the AV-24 are as follows [20]:

Lidar x4: Luminar Iris, max range 275 m, range precision within 2 cm, FoV 120° (H) / 26° (V)

Radar x2: Continental ARS 548 RDI, max range 300–1,500 m, FoV 120°

Camera x6: Allied Vision Mako G-319C, max resolution 2,064 (H) × 1,544 (V), spectral range 300–1,100 nm, max frame rate at full res: 37.6 fps

GNSS (GPS) antenna x4: VectorNav VN-310, position accuracy 1 cm + 1 ppm RMS, heading accuracy 0.1°, position data rate 400 Hz, IMU data rate 800 Hz. PointOne Navigation Polaris RTK Network

Networking/Comms: Cisco IE3300, ports 2×10G, switching bandwidth 128 Gbps. Marelli SA130 telemetry, AVI LTE 9111 modem

Onboard Edge Computing: dSpace Autera AutoBox, Intel Xeon D-2166NT CPU 3 GHz, Nvidia RTX A5000 GPU, 14 TB storage

Drive-by-wire: New Eagle Raptor, IAC custom drive/steer/brake-by-wire systems, MoTeC ECU

1.4 Challenges of Autonomous Racing

High-speed autonomous racing, particularly within benchmark competitions like the Indy Autonomous Challenge [20, 3], represents an exceptionally demanding frontier for robotic control, necessitating navigation of extreme operational requirements distinct from conventional autonomous driving. Core challenges intrinsic to this domain include:

- **Operation at Non-Linear Dynamic Limits:** Vehicles are consistently piloted at the precipice of tire adhesion, entailing highly non-linear vehicle dynamics, significant aerodynamic interactions, and complex tire thermal and wear phenomena. Achieving competitive performance demands precise, aggressive control inputs to exploit these near-limit states without inducing instability.
- **Stringent Spatiotemporal Constraints:** The extreme velocities, often exceeding 270 km/h, impose intense real-time demands on the entire perception-planning-actuation cycle. Millisecond decision latencies and high-frequency control are critical, as minimal deviations or delays can precipitate catastrophic failures or substantial performance degradation.

- **Robust Generalization to Novel Environments:** Autonomous racecars must exhibit robust, often zero-shot or few-shot, generalization to unseen track geometries, diverse surface characteristics, and varying ambient conditions. The pronounced sim-to-real gap in accurately modeling extreme vehicle dynamics further complicates the development of universally effective controllers without extensive, track-specific recalibration, a challenge that techniques like domain randomization [31] aim to mitigate.

These inherent complexities, extreme dynamics, relentless real-time pressures, and the critical demand for high adaptability highlight the profound difficulty in engineering controllers that are simultaneously performant, reliable, and generalizable for this specialized and high-stakes application.

1.5 Learning to Race

This work introduces a learning-based imitation framework designed to harness the strengths of expert controllers while transcending their inherent limitations, particularly within the demanding context of high-speed autonomous racing. Our core strategy involves behavior cloning (BC) [26] to distill the control policies of a high-performance Model Predictive Controller (MPC) into an efficient and powerful transformer policy [2]. While MPC demonstrates proficiency in deriving dynamically feasible trajectories through online constrained optimization based on an explicit dynamics model [23, 3], its practical application confronts several hurdles. Beyond the significant computational overhead, which can be prohibitive for low-latency control loops, MPC performance is fundamentally contingent upon the fidelity of its internal model and the meticulous, often intricate, tuning of its cost function. These factors can circumscribe its adaptability to unmodeled dynamics or rapidly evolving environmental conditions.

Our objective is thus multi-faceted: we aim to transfer the MPC's control capabilities into a lightweight, real-time differentiable neural network that not only executes swiftly on embedded hardware but also offers broader operational advantages. The cloning of the MPC is intended to create a policy exhibiting robust generalization across diverse, previously unencountered racetracks with minimal performance degradation: a critical capability amplified by learning from the vast and varied datasets generated by our methodology. Furthermore, the transformer architecture, with its inherent capacity for sequence modeling [12, 24], can learn to implicitly capture complex temporal dependencies and nuanced vehicle-environment interactions that may be arduous or computationally expensive to encode explicitly within an MPC's mathematical formulation.

While supervised behavior cloning forms the foundational learning stage, our resultant policy offers a potent initialization framework for downstream reinforcement learning (RL) [15, 29]. It is here that the full power of the learned representation can potentially be unlocked. A transformer policy pre-trained via BC provides a crucial baseline, addressing RL's notorious sample inefficiency and often unsafe exploration challenges, particularly in safety-critical domains [30]. More importantly, it equips the RL agent to explore beyond the expert MPC's locally optimal solutions, which frequently arise from its reliance on a fixed model and predefined cost structure. The RL agent, leveraging the transformer's ability to integrate global track context and make decisions over extended horizons, can learn policies that are not merely reactive but strategically proactive. Through direct interaction, even in simulation, the RL fine-tuning process can refine the policy to adeptly handle situations and dynamics implicitly understood by the neural network but not explicitly managed or optimally addressed by the original MPC's constrained optimization problem. This synergistic approach aims to produce control strategies that are more adaptive, robust, and ultimately capable of superhuman capabilities, surpassing even the performance of human experts.

Chapter 2

Related Work

2.1 Autonomous Car Control

The challenge of controlling high-performance vehicles at their handling limits has been tackled using various approaches. Classical methods such as PID controllers and pure pursuit [10] provide foundational control but often struggle with optimality near physical limits. Optimal control methods, particularly MPC [3], have proven highly effective by incorporating vehicle dynamics and constraints directly into the control optimization, enabling high-performance trajectory tracking in cars like the AV-24. However, the accuracy of the dynamics online is a significant bottleneck, especially as parameters can be time-varying. MPC thus requires reparameterization and cost function tuning for new environments, tracks, and conditions to reach optimal performance. This is especially apparent when considering the relationship between slip angle and lateral forces when subjected to changing tire temperatures that affect the grip of the car to the ground. As more dynamics constraints are layered into the MPC, like bank, grade, lateral loading, longitudinal loading, slip angle, and tire temperature, more approximation is required in order to run the controller online in real time. The large parameter space required to properly model these dynamics is also impractical to accurately model, especially as the relationship between these parameters is highly nonlinear. This is only further complicated as properties like wind speed, tire degradation, tire alignment, and outdoor temperatures can drastically change the relationship between them.

2.2 Transformers in Control

Originally revolutionizing natural language processing [2], transformer architectures have increasingly been adapted for sequential decision-making tasks, including reinforcement learning and control. Their self-attention mechanism allows them to effectively weigh the importance of different elements in long input sequences, making them suitable for capturing temporal dependencies in system states and actions. Works like [12, 24] have shown the potential of transformers to model behavior and act as policies in various domains. More recently, we've seen transformers successfully control high degree of freedom humanoids in the real world in [27]. Our work applies this architecture specifically to the domain of continuous control for high-speed autonomous racing, leveraging its sequence modeling capabilities to behavior clone an expert MPC policy.

2.3 Behavior Cloning and Reinforcement Learning

Imitation learning, typically framed as behavior cloning (BC), trains a policy to regress expert actions given observed states. For on-road driving, this dates back to ALVINN [26], but modern work on regular day-to-day cars uses deep networks to clone richly labeled expert trajectories [9]. BC enjoys stability and sample efficiency, yet is bounded by the demonstrator's performance and suffers from covariate shift: at test time, the learner encounters states it has never seen. DAgger addresses this by iteratively querying the expert on the learner's states and adding those samples to the training set [30]. BC approaches like this have enabled real-time neural-network control when the available compute isn't sufficient to run MPC at the required speed, such as in high-speed drone navigation.

[1]. BC also enables mimicry of human-experts as in the previously mentioned [27] where human pose estimation is used to train a humanoid to walk.

Reinforcement learning (RL) can, in principle, exceed the expert, but is notoriously sample inefficient on real systems. A common recipe, therefore, initializes the policy with BC and fine-tunes with RL. In [17, 18], we've seen how RL training in simulation, combined with pretrained behavior cloned models, can create policies that exceed available expert policies.

Recently, another paradigm of learning has arisen, RL in simulation and zero or few-shot transfer to real. The strategy has enabled champion-level quadrotor racing [13] and quadruped locomotion in zero-shot sim to real environments.

Chapter 3

Methodology

Our methodology consists of three stages:

- Generating diverse and realistic racetracks in simulation generated by GPS data from real race tracks internationally.
- Computing a set of optimal reference race lines for each track and collecting data of an expert MPC controller formulated for the AV-24 driving these race lines.
- Training a transformer with a supervised objective to behavior clone the trajectories executed by the expert MPC.

3.1 Racetrack Generalization and Representation

A major concern of this work was minimizing domain shifts from racetrack to racetrack. Since autonomous races are commonly hosted on racetracks that the car has never encountered before, and preparation time before races may only consist of a couple of practice sessions, it is critical that the car performs zero-shot on unseen tracks. Another concern with training on a single track in simulation is the effects of compounding inaccuracies present in the simulated version of the racetrack. This may cause a racetrack in simulation to effectively act as a separate track as compared to its real counterpart.

To address this, we instead develop a procedural pipeline to generate over 100 unique race tracks in simulation from OpenStreetMap data. To acquire this dataset, we:

- **Track GPS Point Identification**: We identified real-world racetracks globally using OSM queries based on GPS coordinates or relevant tags (e.g., `highway=track`, `sport=racing`). This gave us sparse GPS points defined both the main track and irrelevant side roads and access roads surrounding the racetracks as seen in Figure 3.1

- **Track Isolation and Extraction**: The raw track waypoints were then interpolated to increase resolution to create complete paths. They were then segmented to identify distinct track sections as many tracks offer several different track configurations to offer different race formats. Finally a random walk algorithm was used extract the largest primary closed racing loop from potentially several possible racetrack configurations. This multi-step processing pipeline can be seen in 3.2.
- **Track Representation**: For each track, we used the interpolated GPS points to compute a smooth centerline representation along with corresponding inner and outer track boundaries, defining the drivable area.

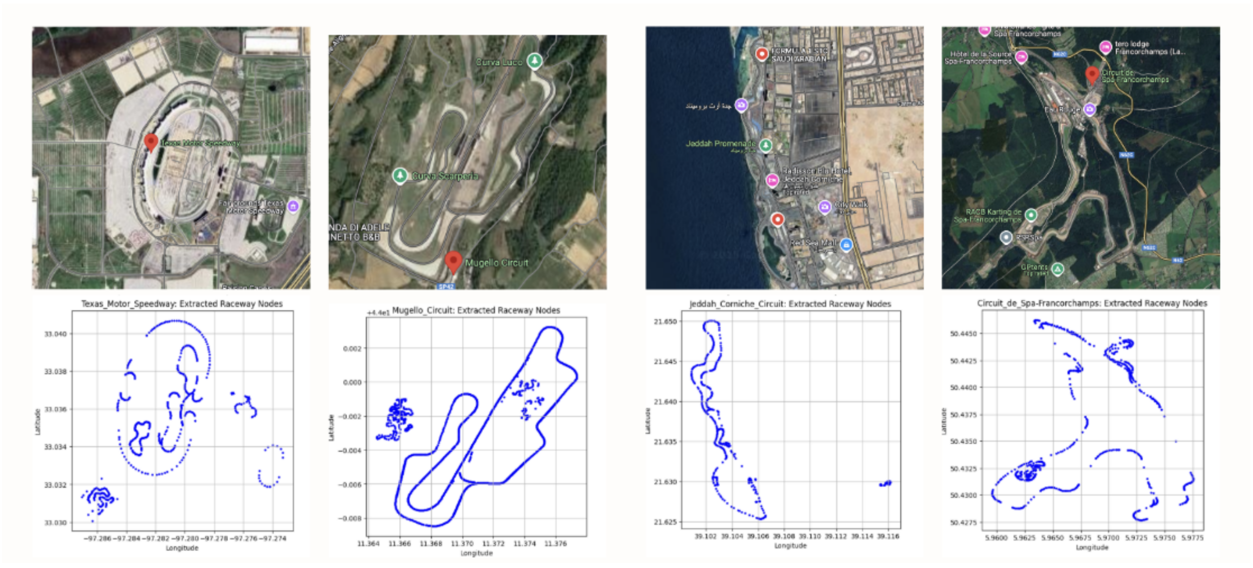

Texas_Motor_Speedway: Extracted Raceway Nodes
Mugello_Circuit: Extracted Raceway Nodes
Jeddah_Corniche_Circuit: Extracted Raceway Nodes
Circuit_de_Spa-Francorchamps: Extracted Raceway Nodes
Latitude
Longitude

Figure 3.1: Extracted nodes from OpenStreetMap data for sample tracks (Texas Motor Speedway, Mugello Circuit, Jeddah Corniche Circuit, Spa-Francorchamps shown).

3.2 Offline Trajectory Optimization

For each generated track, an optimal racing line is computed to serve as the reference trajectory for the MPC controller. We employ a spline-based minimum-curvature trajectory optimization technique presented in [16]. This method optimizes the control points of a spline representing the vehicle's path to minimize overall curvature, which correlates strongly with minimum lap time, while adhering to vehicle dynamics and track boundary constraints.

Vehicle Model: The optimization utilizes a standard double-track vehicle model as seen in 3.3, capturing dynamics including lateral and longitudinal forces at each tire.

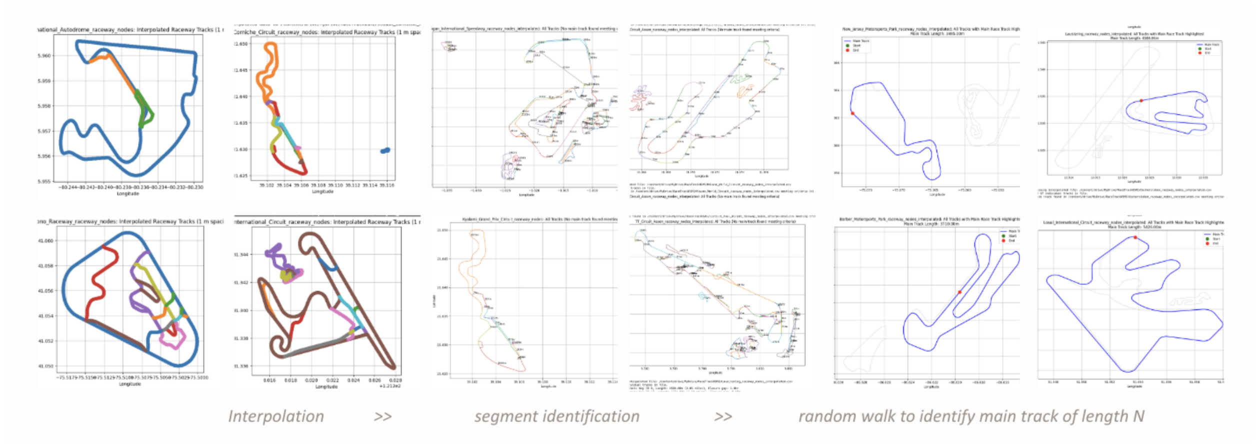


Figure 3.2: Processed track segments derived from the extracted OSM nodes, showing the interpolation, segment identification, and random walk for main track isolation processing steps.

Optimization Formulation: The non-linear programming problem aims to minimize the total traversal time ($\min \sum t_i$) subject to:

- **Decision Variables:** Vehicle state $x_i = [p_x, p_y, \psi, \omega, v, \beta]^T \in \mathbb{R}^6$ (including position, orientation, velocity, slip angle), control inputs $u_i = [F_d, F_b, \delta]^T \in \mathbb{R}^3$ (drive/brake forces, steering angle), and segment time t_i.
- **Dynamics Constraints**: $x_{i+1} = f(x_i, u_i, t_i)$, enforcing that state transitions respect the vehicle's dynamic model f.
- **Path and Input Constraints**: State constraints $x_i \in \mathcal{X}$ ensure the vehicle remains within track boundaries ($d_{right} \leq p_{y,i} \leq d_{left}$), while input constraints $u_i \in \mathcal{U}$ respect actuator limits (steering range, max acceleration/braking) and physical limits (tire friction ellipse, engine power).

3.3 Expert (MPC) Data Generation

An online model predictive controller (MPC) acts as our expert to behavior clone. We use this MPC to track the set of race lines generated in 3.2.

Observation Space

During MPC data collection and at test time, state observations are generated by an Extended Kalman Filter (EKF) [25]. This EKF takes observations from two GPS units and

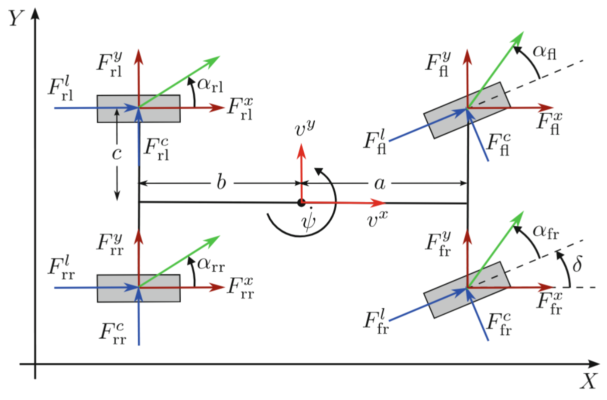


Figure 3.3: Double Track Model from [23]

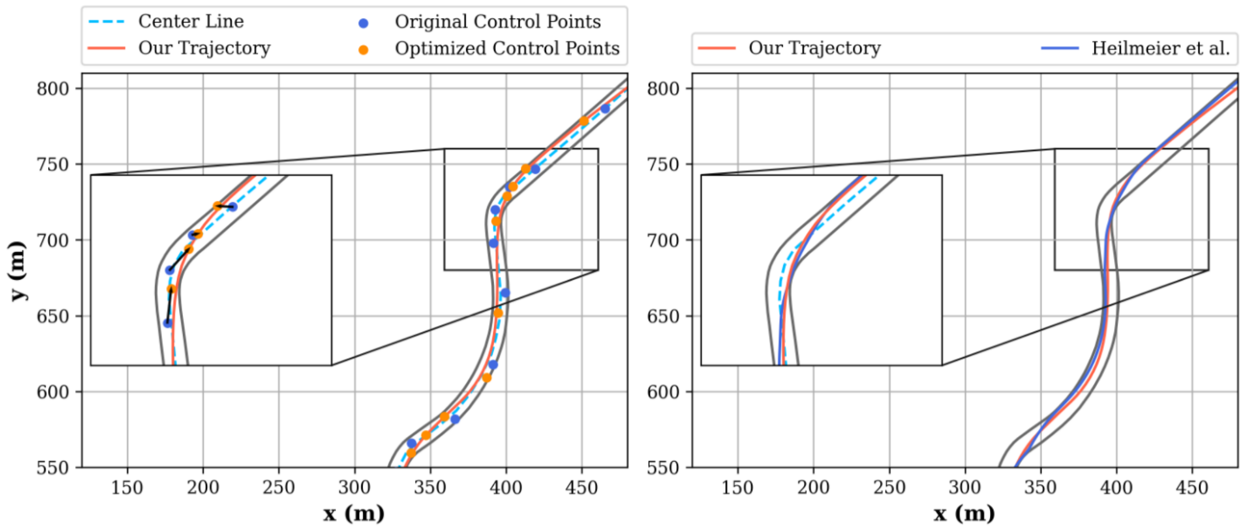


Figure 3.4: Trajectory optimization process (left, middle) from [16] and the vehicle dynamics model (right)

two IMUs to provide state estimation at 200 Hz. The MPC uses the most recent state estimation from the EKF when starting the solve. Since the MPC runs at 30 Hz it outputs control actions every 33 ms. Since MPC takes 33 ms to solve, the state $\vec{x_0}$ received at t_0 is dead reckoned using a bicycle model [28] one state forward in time using fourth-order Runge Kutta integration to estimate $\tilde{\vec{x_1}}$. MPC then uses $\tilde{\vec{x_1}}$ to schedule a solve for the action a_1 which is applied at t_0.

MPC Formulation

3.4 Expert MPC Formulation

The expert controller we behavior clone from is a Model Predictive Controller (MPC) that optimizes steering, drive, and brake forces over a finite horizon while respecting a vehicle model and tire friction limits. This MPC is parameterized about the centerline of the racetrack in the Frenet-Serret frame.

Optimization Problem

At each control step, the MPC solves

$$\min_{\mathbf{u}_{0:N-1}} \sum_{k=1}^{N} \left\| \begin{bmatrix} x_{k,x} - x_{\mathrm{ref},k} \\ x_{k,y} - y_{\mathrm{ref},k} \end{bmatrix} \right\|_Q^2 + \sum_{k=1}^{N-1} \left\| \frac{d_k - d_{k-1}}{\Delta\delta_k} \right\|_R^2 \tag{3.1}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = f_{\mathrm{dyn}}(\mathbf{x}_k, \mathbf{u}_k), \qquad k = 0, \dots, N-1, \tag{3.2}$$

$$\mathbf{x}_0 = \mathbf{x}(t), \tag{3.3}$$

$$\delta_{\min} \le \delta_k \le \delta_{\max}, \tag{3.4}$$

$$\Delta\delta_{\min} \le \Delta\delta_k \le \Delta\delta_{\max}, \tag{3.5}$$

Here $\mathbf{x}_k = \begin{bmatrix} p_x & p_y & \psi & v_x & v_y & \omega & \delta \end{bmatrix}^\top$ is the state, $\mathbf{u}_k = \begin{bmatrix} \Delta\delta & F_{\mathrm{drive}} & F_{\mathrm{brake}} \end{bmatrix}^\top$ the control input, d_k the current steering command, and $Q, R \succ 0$ weighting matrices. The horizon length is N with sample time $T_s = 33\,\mathrm{ms}$.

Tire Dynamics

The dynamics f_{dyn} in 3.2 are represented using the Pacejka non-linear tire model in the body frame [8]. The tire forces in the x direction are computed as:

$$F_{x,fj} = \tfrac{1}{2}\, k_{\text{drive}}\, F_{\text{drive}} + \tfrac{1}{2}\, k_{\text{brake}}\, F_{\text{brake}} - \tfrac{1}{2}\, f_r\, m\, g\, \frac{l_r}{l}, \tag{3.6}$$

$$F_{x,rj} = \tfrac{1}{2}\, (1 - k_{\text{drive}})\, F_{\text{drive}} + \tfrac{1}{2}\, (1 - k_{\text{brake}})\, F_{\text{brake}} - \tfrac{1}{2}\, f_r\, m\, g\, \frac{l_f}{l}. \tag{3.7}$$

while the tire forces in the y direction are computed as:

$$F_{z,\text{fl/fr}} = \tfrac{1}{2}\, mg\, \frac{l_r}{l_f + l_r} \; - \; \tfrac{1}{2}\, \frac{h_{\text{cog}}}{l_f + l_r}\, ma_x \; \pm \; k_{\text{roll}}\Gamma_y \; + \; \tfrac{1}{4}\, c_{l,f}\, \rho A v^2, \tag{3.8}$$

$$F_{z,\text{rl/rr}} = \tfrac{1}{2}\, mg\, \frac{l_f}{l_f + l_r} \; + \; \tfrac{1}{2}\, \frac{h_{\text{cog}}}{l_f + l_r}\, ma_x \; \pm \; (1 - k_{\text{roll}})\Gamma_y \; + \; \tfrac{1}{4}\, c_{l,r}\, \rho A v^2, \tag{3.9}$$

$$\Gamma_y = \frac{h_{\text{cog}}}{\tfrac{1}{2}\big(t_{w_f} + t_{w_r}\big)} \Big[\big(F_{y,\text{rl}} + F_{y,\text{rr}}\big) + \big(F_{x,\text{fl}} + F_{x,\text{fr}}\big) \sin\delta + \big(F_{y,\text{fl}} + F_{y,\text{fr}}\big) \cos\delta \Big]. \tag{3.10}$$

where Γ_y is zero when solving the MPC online due to the use of a single-track model. The tire forces in the z direction are computed as:

$$F_{y,\text{fl}} = \mu\, F_{z,\text{fl}}\; \sin\!\Big(C_f\; \arctan\!\big(B_f\, \alpha_{\text{fl}}\big)\Big), \tag{3.11}$$

$$F_{y,\text{rl}} = \mu\, F_{z,\text{rl}}\; \sin\!\Big(C_r\; \arctan\!\big(B_r\, \alpha_{\text{rl}}\big)\Big), \tag{3.12}$$

$$\alpha_{fl/fr} = \delta \; - \; \arctan\!\Big(\frac{l_f\, \omega_z + v \sin\beta}{v \cos\beta \; \pm \; \tfrac{1}{2} t_{wf}\, \omega_z}\Big), \tag{3.13}$$

$$\alpha_{rl/rr} = \arctan\!\Big(\frac{l_r\, \omega_z - v \sin\beta}{v \cos\beta \; \pm \; \tfrac{1}{2} t_{wr}\, \omega_z}\Big). \tag{3.14}$$

Finally, the overall tire friction at each wheel is constrained:

$$\left(\frac{F_{x,ij}}{\mu_{ij} F_{z,ij}}\right)^2 + \left(\frac{F_{y,ij}}{\mu_{ij} F_{z,ij}}\right)^2 \leq 1, \quad \forall\, \text{wheel } ij. \tag{3.15}$$

Vehicle Dynamics

Using the tire model, we generate the nonlinear vehicle dynamics for the MPC [8]:

$$\begin{aligned}
\dot{v} = \frac{1}{m}\Big(&(F_{x,rl} + F_{x,rr})\cos\beta + (F_{x,fl} + F_{x,fr})\cos(\delta - \beta) + (F_{y,rl} + F_{y,rr})\sin\beta \\
&- (F_{y,fl} + F_{y,fr})\sin(\delta - \beta) - \tfrac{1}{2}\, c_d\, \rho\, A\, v^2 \cos\beta\Big), \qquad (3.16)
\end{aligned}$$

$$\begin{aligned}
\dot{\omega}_z = \frac{1}{J_{zz}}\Big[&(F_{x,rr} - F_{x,rl})\,\frac{tw_{,r}}{2} - (F_{y,rl} + F_{y,rr})\, l_r \\
&+ (F_{x,fr} - F_{x,fl})\, \cos(\delta)\,\frac{tw_f}{2} + (F_{y,fl} - F_{y,fr})\, \sin(\delta)\,\frac{tw_f}{2} \\
&+ (F_{y,fl} + F_{y,fr})\, \cos(\delta)\, l_f + (F_{x,fl} + F_{x,fr})\, \sin(\delta)\, l_f\Big]. \qquad (3.17)
\end{aligned}$$

We then discretize these dynamics with Runge Kutta fourth-order integration and linearize about the trajectory every time step. This defines our state evolution constraint in the MPC.

Actuator Mapping

The optimizer returns the sequence $\vec{u} = \left[\Delta\delta^\star, F^\star_{\text{drive}}, F^\star_{\text{brake}}\right]^\top$ for the first sample, which is converted to physical commands using three output modules:

1. **Steering**: the desired wheel angle $\delta^\star = \delta_{k-1} + \Delta\delta^\star$ is sent to a Raptor embedded PID that closes the inner steering-motor torque loop and outputs motor torque $\tau_{\text{steer}}\,[\mathrm{N\,m}]$.
2. **Drive**: the required longitudinal drive force $F^\star_{\text{drive}}\,[\mathrm{kN}]$ is fed into a look-up engine torque map $\tau_{\text{eng}} = \text{map}(F^\star_{\text{drive}},\, v_x)$, which is normalized to a throttle percentage $u_{\text{throt}} \in [0, 1]$.
3. **Brake**: the commanded brake force $F^\star_{\text{brake}}$ passes through a static brake model $p_{\text{brake}} = \text{map}(F^\star_{\text{brake}})$ that produces hydraulic pressure in kPa.

The complete MPC therefore delivers τ_{steer}, u_{throt} and p_{brake} at 30 Hz, which are applied by the drive-by-wire system one sample later after latency compensation. Note for our BC training data, we clone $\vec{u}$ directly instead of τ_{steer}, u_{throt} and p_{brake}.

State Randomization

One of the core challenges in behavior cloning is the distributional shift between the expert data and the learner's rollouts. During training, the learner only sees the optimal states the expert generated. However, at test time, the learner's own actions influence the state distribution it encounters. Small prediction errors the learner makes compound over time, causing the model to drift into states not seen during training. Since the model is now out of distribution it often takes increasingly poor actions, leading to trajectory divergence or, in this case, a crash. Out of distribution sampling is especially problematic in high-speed and safety-critical domains like autonomous racing, where small deviations can lead to catastrophic crashes in less than a second.

To address this, we apply a technique similar to DAgger [30]. After saving the current state action pair into the dataset, we inject Gaussian noise into the control action before executing it in the simulator. This forces the expert to recover from perturbed states that it would not normally enter otherwise. By including these recovery trajectories in the training dataset, we broaden the distribution of states the learner is exposed to, improving the learner's robustness to error.

Velocity Randomization

During racing and testing, the car is constantly accelerating and decelerating through turns. To prepare the policy for all these scenarios, during data collection, we randomly vary the maximum desired velocity from 30 mp/h to 170 mp/h (∼40 km/h to 270 km/h). We note that at speeds lower than 30 mp/h, the AV-24 uses pure pursuit rather than MPC, so the policy is not trained at these speeds to prevent two separate modes from occurring in the dataset.

Behavior Cloning

The core of our approach is training a transformer model to behavior clone the actions generated by the expert MPC.

Tokens

In our formulation, each time step i is represented by a single token. To create this token, we take as input both the state $\vec{s_i}$ and $\vec{r_i}$, a look-ahead of the next M steps, where:

$$\vec{s_i} = \left[e_y, e_\psi, v_x, v_y, \dot{\psi}\right]^T \in \mathbb{R}^5 \tag{3.18}$$

$$\vec{r_i} = \left[v_x, v_y, \psi\right]^T \in \mathbb{R}^{3M} \tag{3.19}$$

Here, $\vec{s_i}$ and $\vec{r_i}$ are in the Frenet-Serret frame of the race line. Note that $\vec{s_i}$ is a subset of the state x_i MPC solves with. Specifically, e_y is lateral error relative to the race line, e_ψ is yaw error relative to the race line, v_x is longitudinal velocity, v_y is lateral velocity, and $\dot{\psi}$ is yaw rate. We note that to match MPC, $\vec{s_i}$ is dead-reckoned one state forward in time using fourth-order Runge Kutta integration to acquire $\tilde{\vec{s}}_{i+1}$ as we are solving for the action in the next time step. We then learn a linear projection for $\tilde{\vec{s}}_{i+1}$ and $\vec{r_i}$ separately and concatenate them:

$$h_i^s = W_s \tilde{\vec{s}}_{i+1} \in \mathbb{R}^{d_s} \tag{3.20}$$

$$h_i^r = W_r \vec{r_i} \in \mathbb{R}^{d_r} \tag{3.21}$$

$$t_i = \text{concat}(h_i^s, h_i^r) \in \mathbb{R}^{d} \tag{3.22}$$

where W_s is the learned projection for $\tilde{\vec{s}}_{i+1}$, W_r is the learned projection for $\vec{r_i}$, t_i represents the token at time i, and $d = d_s + d_r$. W_s and W_r are shared across time steps. Note this state-only version of the transformer is only conditioned on the state and reference line from 3.20 and 3.21.

A state-action version of the transformer is also implemented where tokens from previous time steps have their respective actions appended to them:

$$h_{i-1}^s = W_s \tilde{\vec{s}}_{i+1} \in \mathbb{R}^{d_s} \tag{3.23}$$

$$h_{i-1}^r = W_r \vec{r_{i-1}} \in \mathbb{R}^{d_r} \tag{3.24}$$

$$h_{i-1}^a = W_a \vec{a_{i-1}} \in \mathbb{R}^{d_a} \tag{3.25}$$

$$t_{i-1} = \text{concat}(h_{i-1}^s, h_{i-1}^r, h_{i-1}^a) \in \mathbb{R}^{d} \tag{3.26}$$

where $d = d_s + d_r + d_s$. Since we have not yet taken the action we are predicting at the current time step, we fill the current action with $\vec{0}$ to keep the dimension of the token uniform.

$$t_i = \text{concat}(h_i^s, h_i^r, \vec{0}) \in \mathbb{R}^{d} \tag{3.27}$$

At test time, we replace the $\vec{0}$ with a_{i-1} after applying the action as described in 3.26. The transformer keeps a context window of the last N tokens, allowing the attention mechanism to see the state evolution over time.

Architecture

The transformer uses a standard decoder-only architecture with L layers, each comprising a multi-head self-attention mechanism with causal masking over the tokens, residual connections, a position-wise feedforward MLP, and layer normalization as follows:

$$\tilde{H}_l = \text{LayerNorm}(H_l) \tag{3.28}$$

$$\tilde{H}_l = \tilde{H}_l + \text{MHSA}(\tilde{H}_l) \tag{3.29}$$

$$H_{l+1} = \tilde{H}_l + \text{MLP}(\text{LayerNorm}(\tilde{H}_l)) \tag{3.30}$$

where H_l is the input to layer l and H_{l+1} is the output to the next layer. Note here we use pre-layer normalization as described in [32]. Finally, at the output of the last layer L of the transformer, we take the last token in the sequence and feed it into a feedforward MLP to predict the action at time i:

$$a_i = \left[f, \delta\right]^T \in \mathbb{R}^2 \tag{3.31}$$

where f represents longitudinal drive force and δ represents steering.

Multi-Time Step Prediction

An alternate formulation of the model predicts $k + 1$ time steps into the future. So instead of predicting a_{i+1} from s_i where k here is zero, the model can instead predict a_{i+1+k} from s_i. This becomes relevant when dealing with inference latency as discussed in 4.4 and 5.1.

Test Time Pipeline

At test time in simulation and on the real on-board hardware, several steps compose the inference process. First, just like how MPC dead-reckons the state in 3.3, the state is dead-reckoned from s_i to $\tilde{\vec{s}}_{i+1}$. This state is then communicated over the robotic middleware [22] from the dead reckoning process to the model inference process. Finally, the inference process makes a prediction for the next time step, a_1, and communicates the action to the hardware to execute. Note that this entire process must occur in 33 ms in order to make the execution window to apply the action in time for the next time step.

Chapter 4

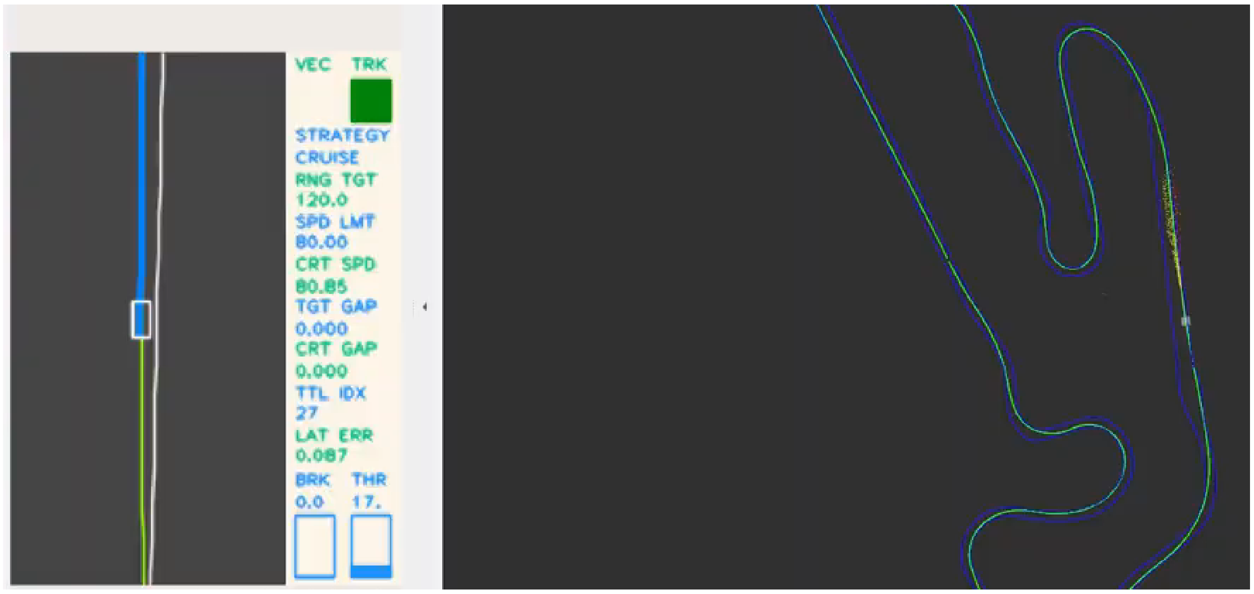
Experiments

We design our experiments to test the performance of the trained model in both unseen simulation environments and zero-shot in the real world. The evaluations in both simulation and the real world are conducted on Las Vegas Road Course (LVRC).

4.1 Simulation Environment

Simulations are conducted using LGSVL simulator [14], an open-source platform for autonomous vehicle simulation. The environment utilizes a high-fidelity dynamics model for the racecar tuned specifically for the autonomous platform. This model captures key vehicle dynamics, including tire forces and suspension behavior, providing a realistic testbed.

Figure 4.1: LGSVL Simulator



VEC TRK

STRATEGY CRUISE

RNG TGT 120.0

SPD LMT 80.00

CRT SPD 80.85

TGT GAP 0.000

CRT GAP 0.000

TTL IDX 27

LAT ERR 0.087

BRK 0.0

THR 17.

Figure 4.2: LGSVL Simulator Map View

4.2 Baseline

The primary performance baseline is the expert MPC controller used for generating the training data. Its performance represents the target behavior for the behavior cloned policy. We also include a pure pursuit controller [10], which is used as a safety controller in case the transformer and MPC both fail.

4.3 Zero-Shot Simulation

Race Line Tracking Performance Quantitative results are summarized in Table 4.1. At speeds up to 120 mp/h, the transformer policy achieves average lateral tracking errors that resemble the expert MPC baseline. This demonstrates successful behavior cloning.

Imitation Quality The mean squared error metrics reported in Table 4.2 quantify how closely the transformer mimics the MPC actions during simulation runs time step by time step, given states the transformer drives the car into. While average losses are generally low across speeds, reflecting good overall imitation, the peak losses, particularly for longitudinal force, tend to increase at speeds exceeding 100 mp/h. This suggests the policy struggles to replicate the expert's precise and aggressive actions during challenging, high-speed maneuvers, correlating with the observed increase in tracking errors. We note that these high lateral errors are seen mainly in hairpin turns, where the learned model tends to oversmooth the steering commands compared to the abrupt steering command of the MPC.

Table 4.1: Lateral-tracking error (meters) on the unseen LVRC track in simulation. Comparing transformer (MODEL) vs expert (MPC).

Speed	**Run**	**MODEL (m)**		**MPC (m)**	
		Avg	Peak	Avg	Peak
60 mp/h	1	0.1738	0.3210	0.1040	0.4160
	2	0.1553	0.4667	0.3224	1.2890
80 mp/h	1	0.1480	0.5927	0.1285	0.5141
	2	0.1343	0.5376	0.1380	0.5521
100 mp/h	1	0.2105	1.0533	0.1378	0.5513
	2	0.2105	1.0538	0.3838	1.1522
120 mp/h	1	0.2258	1.3278	0.1459	0.5979
	2	0.2295	1.3250	0.1456	1.4193

Table 4.2: Imitation-loss metrics (MSE) comparing transformer output to MPC targets on unseen LVRC in simulation

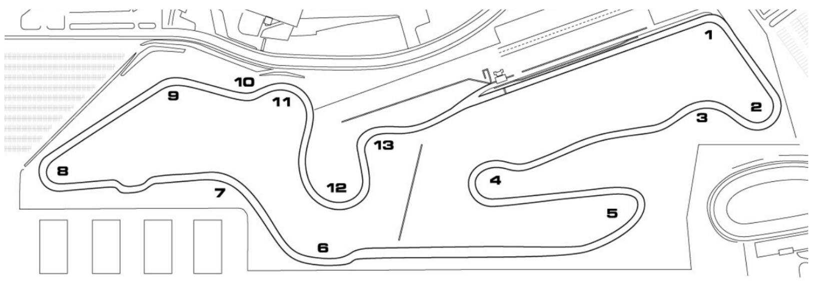
Speed	**Metric**	**Run 1**	**Run 2**
60 mp/h	Avg f	1.0294	1.060
	Peak f	5.9347	6.112
	Avg δ	0.0040	0.0042
	Peak δ	0.0252	0.0260
80 mp/h	Avg f	0.3007	0.309
	Peak f	1.7968	1.850
	Avg δ	0.0025	0.0026
	Peak δ	0.0152	0.0157
100 mp/h	Avg f	0.7761	0.799
	Peak f	7.0871	7.300
	Avg δ	0.0030	0.0031
	Peak δ	0.0331	0.0341
120 mp/h	Avg f	0.8178	0.842
	Peak f	4.2437	4.371
	Avg δ	0.0035	0.0037
	Peak δ	0.0414	0.0427

4.4 Zero-Shot Simulation to Real

We tested the learned policy on the real-world LVRC pictured in 4.3b. The turns and map of the track are depicted in 4.3a. Note the policy deployed in real was trained on LVRC in simulation, unlike the model in the sim-to-sim experiment in 4.3. Due to the prohibitive cost of professional race tracks, typically measured in tens of thousands of dollars per day, we were only able to conduct limited experiments in real, which, as we will outline here and in 5.1 greatly limited results.

We note in the real experiment plots that the learned model is in control of the car and therefore dictating the state evolution. The MPC actions are a point-by-point reaction to the state evolution being created by the learned model. Another point to note is that on the AV-24, there is a constant steering offset that reflects the misalignment of the wheels relative to the steering column. To handle this mismatch, all control outputs, regardless of the controller, have a constant steering bias applied to them before being actuated on the hardware as reflected in 4.5.

In 4.6, we note the learned model allows the car to drive +/- 2 mp/h around the target speed compared to the MPC that prefers to narrowly drive a band exactly at the target speed. This causes the desired longitudinal acceleration to seem drastically different when compared. This emulates the oversmoothing seen in the sim to sim experiment in 4.3.



(a) Track Turns

(b) Aerial Photo

Figure 4.3: Las Vegas Road Course

Predicting The Current Action

For our on track test, we set the max speed to 40 mp/h and speed scale factor to 0.6 while the model predicted the next action, where $k = 0$, as explained in 3.4.

Figure 4.4: AV-24 Driving on Las Vegas Road Coarse

For this experiment, we engage the learned model soon after turn 5, as seen in 4.3a, and disable the model shortly after the chicane that can be seen in 4.3a after turn 7. We note that the learned model clearly emulates a critically underdamped control system as oscillations slowly build up over time in the system while driving on a straight-away. We realized in a later track testing session that this was due to a greater than $3x$ increase in latency. We did not experience this latency in simulation testing, as our benchmarking computers were much more powerful compared to the AV-24 embedded computer.

These oscillations only continued to grow larger in magnitude as the experiment continued. The chicane on LVRC is an S-shaped pair of alternating corners inserted into a straight section of a circuit to force a sharp speed reduction. This chicane, placed after turn 7 in 4.3a forces a quick right turn followed by a left turn. Due to the large magnitude of the oscillations combined with a late application of the brakes on the exit of the chicane, the magnitude of lateral error exceeded 1.5 meters as the car attempted to correct from the late braking on the exit of the chicane. As a safety precaution, the joystick operator observing the racecar applied brakes to bring the car to a stop, as can be seen in 4.8.

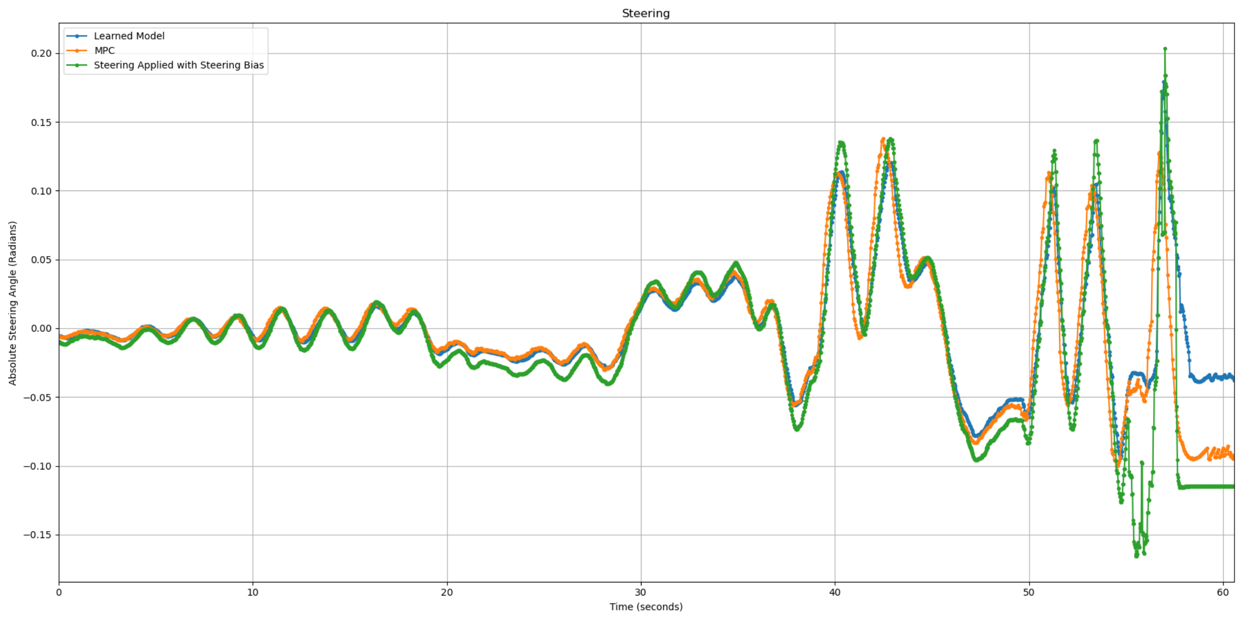


Figure 4.5: Steering: Learned Model vs MPC vs Steering Bias Applied for Real Experiment

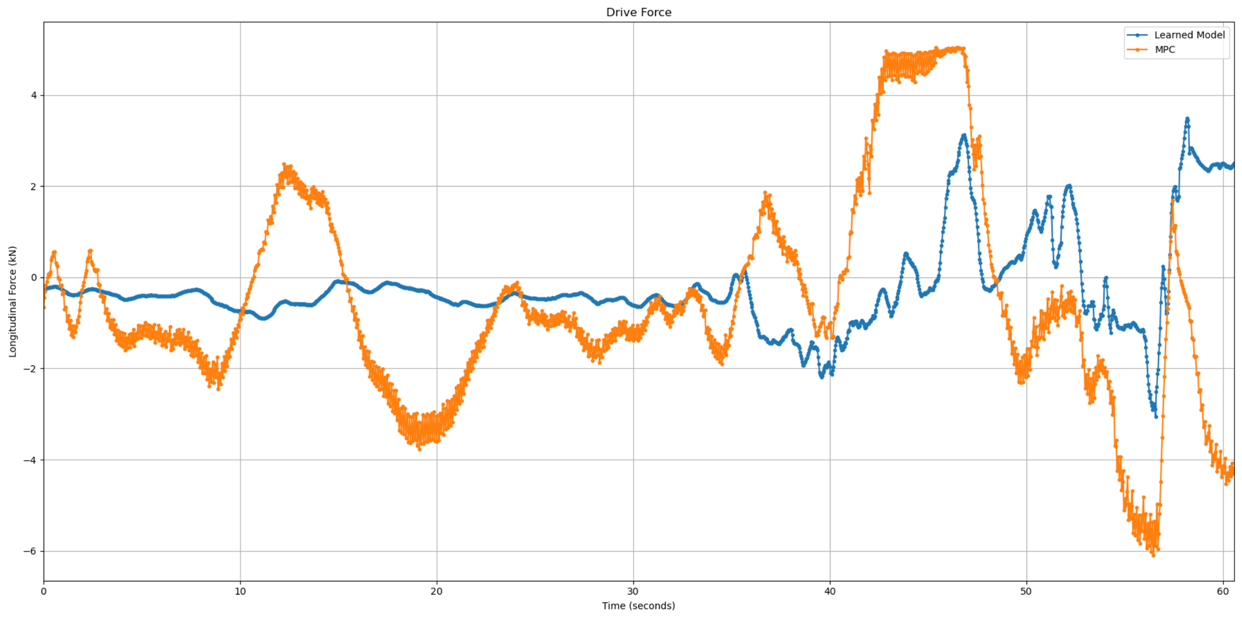


Figure 4.6: Drive Force: Learned Model vs MPC for Real Experiment

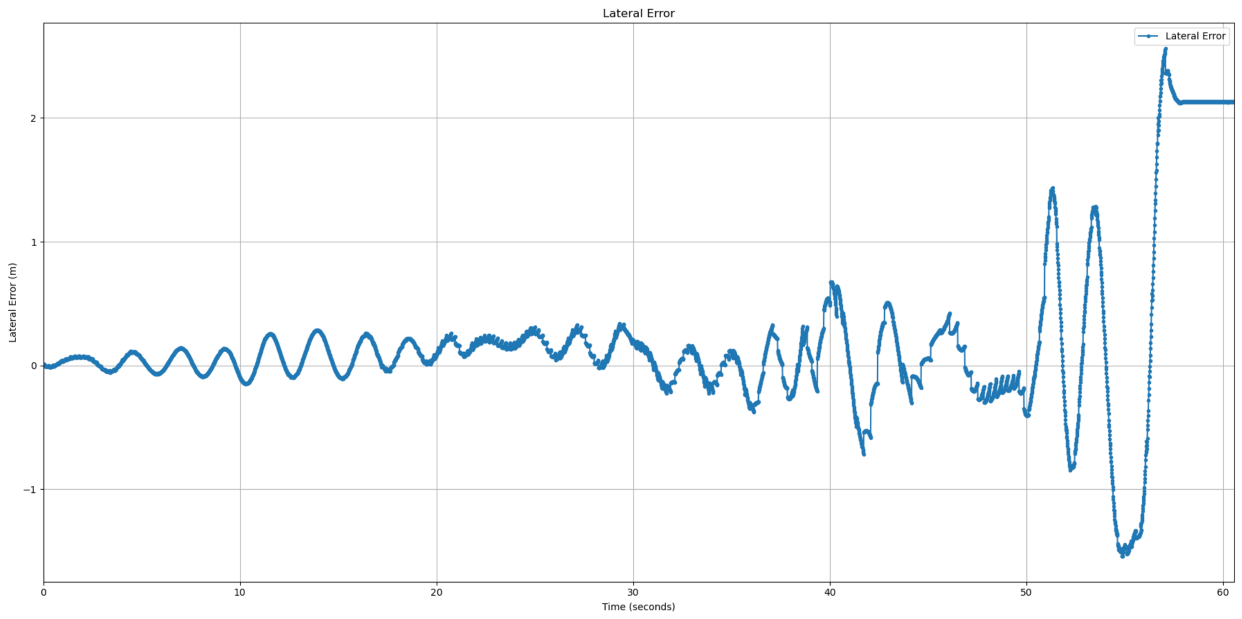


Figure 4.7: Lateral Error for Real Experiment

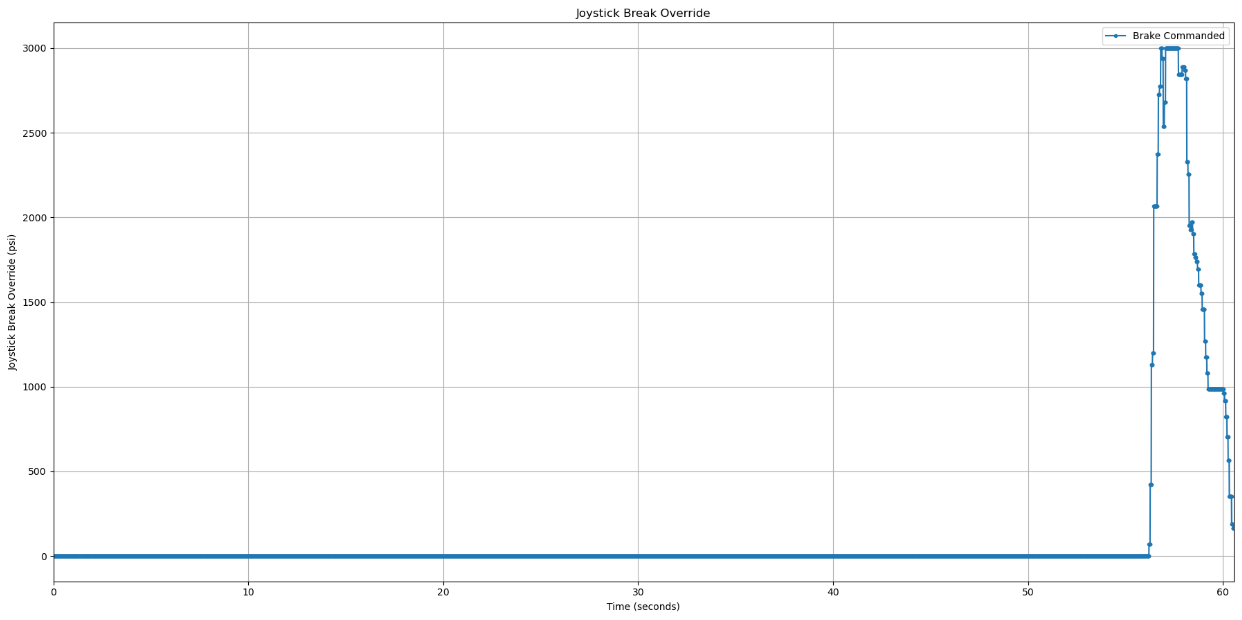


Figure 4.8: Joystick Break Override for Real Experiment

Chapter 5

Discussions

Impact of the Latency

In deployment, we observed a fixed 90 ms command-delivery latency, three times the design budget of one sample ($T_s = 33$ ms).

Even with perfect dead-reckoning, a three-step prediction horizon means the applied control a_{k+3} is computed from a state estimate $\tilde{\vec{s}}_{k+1}$ that is already two integration steps old by the time it reaches the actuators. Any model mismatch is therefore integrated over three samples before feedback can correct it, amplifying process noise and modeling error.

We note that in a later experiment, the model was changed to use the future time step prediction a_{k+1}, and oscillations were reduced enough to take the tight hairpin turn through turn 4.

5.1 Latency Improvements

As previously mentioned, the main limiting factor in our sim-to-real experiments was the latency introduced. This created a phase shift that created oscillations. Latency in the inference pipeline was concentrated in state dead reckoning, middleware communication time, and model inference time. We note that the majority of the latency was due to slow communication in the robotic middleware. To address this, we optimized the pipeline to reduce unnecessary communication, used multiple threads efficiently, and simplified the overall communication structure. We also implemented additional quantization and GPU memory optimization via DMA engine loading.

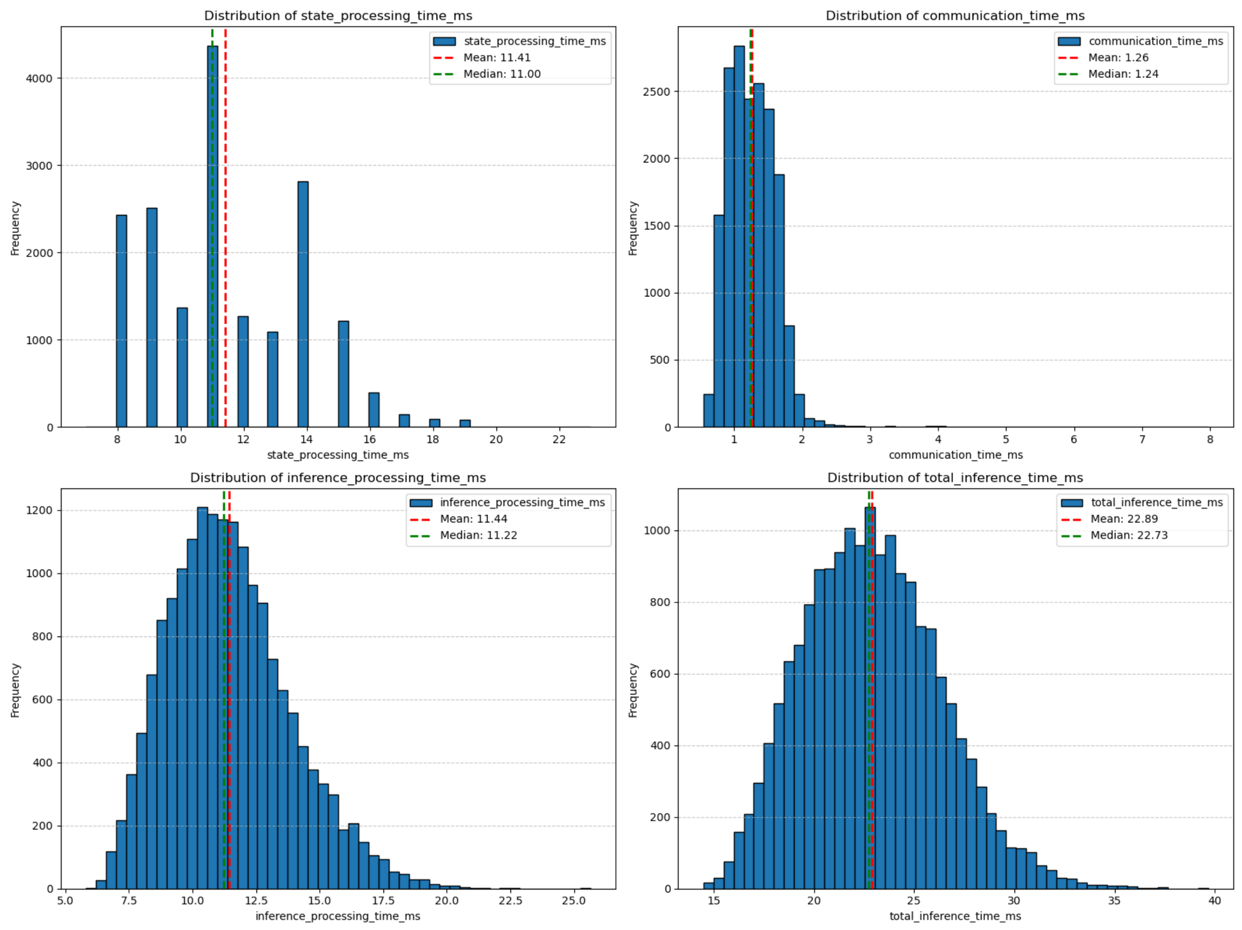


Figure 5.1: Steering: Learned Model vs MPC vs Steering Bias Applied for Real Experiment

5.2 Sim and Real Data Mixture

This work leaves room for improvement in terms of training on not just simulation data, but real-world data collected by the expert MPC. In this work, we avoided utilizing real-world data as our real data from previous track days all use different versions of the MPC controller, which is constantly being updated, tweaked, and improved upon during testing. We were concerned that this would effectively cause the model to learn several different controllers, all representing different modes in the dataset, which may make it hard to identify behavior cloning quality against the single version of the MPC we benchmarked against.

Chapter 6

Conclusion

We built a learned model for zero-shot autonomous racing control and evaluated it on a full-scale autonomous IndyCar. Starting from online GPS data, we generated more than a hundred realistic circuits, computed time-optimal race lines with a physics-based optimizer, and generated expert demonstrations from an expert MPC. A causal transformer was then behavior cloned on this corpus of data and assessed on both held-out simulated tracks and the real Las Vegas Road Course. Across speeds up to 120 mp/h, the policy mimicked the expert's control in simulation and overcame large system delay on the physical car despite perception noise and model mismatch. Detailed ablations highlighted the importance of track diversity, recovery noise, and multi-step prediction for latency compensation. These results position large-scale sequence-model pre-training as a practical alternative to online optimization for high-speed, safety-critical control, and open several avenues: reinforcement learning, domain randomization, self-supervised fine-tuning, and multimodal inputs for closing the remaining gap to expert performance.

Bibliography

[1] René Ranftl Matthias Müller Vladlen Koltun David Scaramuzza Antonio Loquercio1, Elia Kaufmann1. Learning high-speed flight in the wild. *Science Robotics*, 2021.

[2] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Łukasz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer. Attention is all you need. *NeurIPS*, 2017.

[3] Johannes Betz, Tobias Betz, Felix Fent, Maximilian Geisslinger, Alexander Heilmeier, Leonhard Hermansdorfer, Thomas Herrmann, Sebastian Huch, Phillip Karle, Markus Lienkamp, Boris Lohmann, Felix Nobis, Levent Ögretmen, Matthias Rowold, Florian Sauerbeck, Tim Stahl, Rainer Trauth, Frederik Werner, and Alexander Wischnewski. Tum autonomous motorsport: An autonomous racing software for the indy autonomous challenge. *Journal of Field Robotics*, 40(4):783–809, 2023. doi: https://doi.org/10.1002/rob.22153. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22153.

[4] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2024. URL https://arxiv.org/abs/2410.24164.

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. URL https://arxiv.org/abs/2307.15818.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023. URL https://arxiv.org/abs/2212.06817.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[8] Fabian Christ, Alexander Wischnewski, Alexander Heilmeier, and Boris Lohmann and. Time-optimal trajectory planning for a race car considering variable tyre-road friction coefficients. *Vehicle System Dynamics*, 59(4):588–612, 2021. doi: 10.1080/00423114.2019.1704804. URL https://doi.org/10.1080/00423114.2019.1704804.

[9] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4693–4700. IEEE, 2018.

[10] R. Craig Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, Robotics Institute, Carnegie Mellon University, 1992.

[11] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: An embodied multimodal language model, 2023. URL https://arxiv.org/abs/2303.03378.

[12] Antonio Loquercio Matthias Müller Vladlen Koltun Davide Scaramuzza Elia Kaufmann, Leonard Bauersfeld. Decision transformer: Reinforcement learning via sequence modeling. *NeurIPS*, 2021.

[13] Antonio Loquercio Matthias Müller Vladlen Koltun Davide Scaramuzza Elia Kaufmann1, Leonard Bauersfeld. Champion-level drone racing using deep reinforcement learning. *Nature*, 2023.

[14] Hadi Tabatabaee Qiang Lu Steve Lemke Martins Mozeiko Eric Boise Geehoon Uhm Mark Gerow Shalin Mehta Eugene Agafonov Tae Hyung Kim Eric Sterner Keunhae Ushiroda Michael Reyes Dmitry Zelenkovsky Seonman Kim Guodong Rong, Byung Hyun Shin1. Lgsvl simulator: A high fidelity simulator for autonomous driving. *IEEE International Conference on Intelligent Transportation*, 2020.

[15] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[16] John M. Dolan Haoru Xue, Tianwei Yue. Spline-based minimum-curvature trajectory optimization for autonomous racing, 2023.

[17] Bike Zhang Trevor Darrell Jitendra Malik Koushil Sreenath Ilija Radosavovic, Tete Xiao. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 2024.

[18] Trevor Darrell Jitendra Malik Ilija Radosavovic, Sarthak Kamat. Learning humanoid locomotion over challenging terrain, 2024.

[19] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025. URL https://arxiv.org/abs/2504.16054.

[20] ADAS & Autonomous Vehicle International. Indy autonomous challenge, 2024. URL https://adas.mydigitalpublication.com/articles/cover-story-indy-autonomous-challenge?m=71150&i=820187&p=1&article_id=4761453&ver=html5.

[21] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47): eabc5986, 2020. doi: 10.1126/scirobotics.abc5986. URL https://www.science.org/doi/abs/10.1126/scirobotics.abc5986.

[22] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022. doi: 10.1126/scirobotics.abm6074. URL https://www.science.org/doi/abs/10.1126/scirobotics.abm6074.

[23] Milan Vukov Sebastian Sager Mario Zanon, Janick V. Frasch and Moritz Diehl. Model predictive control of autonomous vehicles. In *Lecture Notes in Control and Information Sciences*, volume 455, pages 58–74. Springer, 2014. URL https://doi.org/10.1007/978-3-319-05371-4_3.

[24] Sergey Levine Michael Janner, Qiyang Li. Offline reinforcement learning as one big sequence modeling problem. *NeurIPS*, 2021.

[25] T. Moore and D. Stouch. A generalized extended kalman filter implementation for the robot operating system. In *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.

[26] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[27] Ilija Radosavovic, Bike Zhang, Baifeng Shi, Jathushan Rajasegaran, Sarthak Kamat, Trevor Darrell, Koushil Sreenath, and Jitendra Malik. Humanoid locomotion as next token prediction. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[28] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science Business Media, 2011.

[29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[30] Drew Bagnell Stephane Ross, Geoffrey Gordon. A reduction of imitation learning and structured prediction to no-regret online learning. *Proceedings of Machine Learning Research*, 2011.

[31] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017. doi: 10.1109/IROS.2017.8202133.

[32] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture, 2020. URL https://arxiv.org/abs/2002.04745.