FastRLAP: A System for Learning High-Speed Driving via Deep RL and Autonomous Practicing

Author Names Omitted for Anonymous Review.

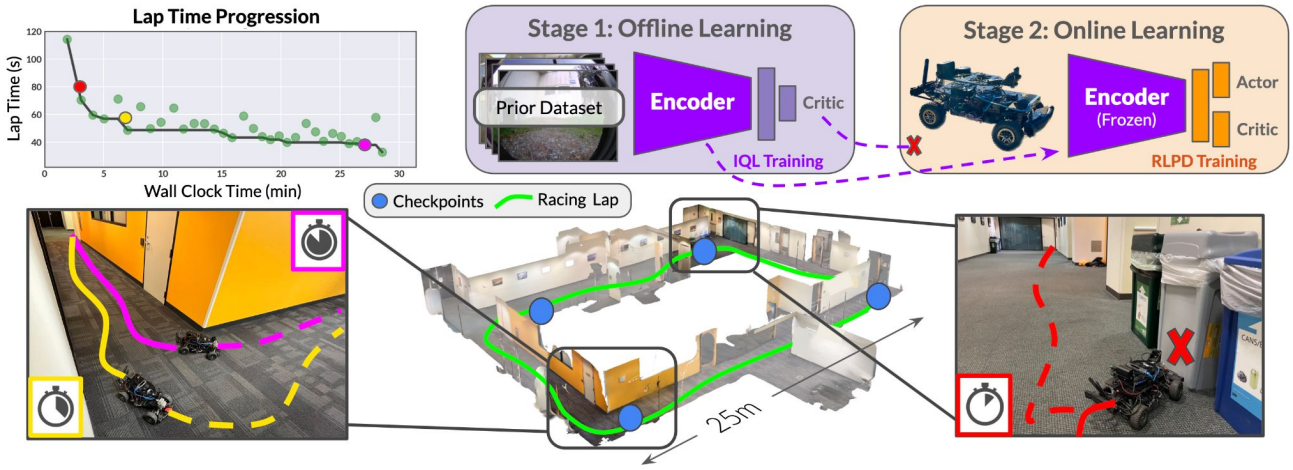

Fig. 1: Fast reinforcement learning via autonomous practicing. By pre-training the RL policy on diverse data (Stage 1), and deploying our autonomous practicing framework for continuous online improvements (Stage 2) in large real-world environments, the robot can autonomously navigate between sparse checkpoints (blue), recovering from collisions during practice (red) and improve its driving behavior to maximize speed (yellow → magenta). FastRLAP can learn aggressive driving comparable to a human expert within 20 minutes of autonomous practice. *Note that the 3D scan is shown for visualization only, and is not available to our system.*

***Abstract*—We present a system that enables an autonomous small-scale RC car to drive aggressively from visual observations using reinforcement learning (RL). Our system, FastRLAP (*faster lap*), trains autonomously in the real world, without human interventions, and without requiring any simulation or expert demonstrations. Our system integrates a number of important components to make this possible: we initialize the representations for the RL policy and value function from a large prior dataset of *other* robots navigating in other environments (at low speed), which provides a navigation-relevant representation. From here, a sample-efficient online RL method uses a single low-speed user-provided demonstration to determine the desired driving course, extracts a set of navigational checkpoints, and autonomously practices driving through these checkpoints, resetting automatically on collision or failure. Perhaps surprisingly, we find that with appropriate initialization and choice of algorithm, our system can learn to drive over a variety of racing courses with less than 20 minutes of online training. The resulting policies exhibit emergent aggressive driving skills, such as timing braking and acceleration around turns, and approach the performance of a human driver using a similar first-person interface.**

I. Introduction

High-speed vision-based navigation presents a range of challenges: aside from the usual difficulties associated with collision-free navigation, it requires controllers that can account for both the vehicle's dynamics and the perceived obstacles (Fig. 4). Learning-based methods offer a particularly appealing way to approach such challenges, as they can directly learn the relationship between perception and vehicle dynamics and in principle capture high-performance driving behaviors. One way that prior work has approached such domains is via imitation learning, acquiring end-to-end skills from expert demonstrations [1, 2]. However, if our aim is to maximize performance, we might instead prefer to directly adapt the navigational strategy to the vehicle *autonomously*. By learning from *autonomous* experience in an environment, reinforcement learning (RL) can enable this, analogously to progress in other domains such as games and robotic manipulation, where policies trained via RL have even exceeded human-level performance [3–6].

But the autonomous setting presents major challenges for RL: unlike other domains, it is impossible to reset the system to a random state, and so the learning process is highly dependent on the system's ability to make continual progress without getting *stuck*. Starting the training process from a randomly-initialized policy and learning solely via trial-and-error would likely result in catastrophic failure. Instead, the RL-based system should train automatically, without supervision, not just improving its policy performance but also smoothly recovering from failures or collisions. The goal of this paper is to address these challenges and understand how RL can be applied to autonomously learn high-speed driving from vision.

To this end, we design a high-performance system for **Fast** **R**einforcement **L**earning via **A**utonomous **P**racticing. FastRLAP (*faster-lap*), which uses an automatic goal curriculum to guide a goal-conditioned RL policy to quickly adapt to the target environment and improve its performance over multiple laps, without requiring human interventions (see Fig. 1). We

then leverage data-efficient online RL, bootstrapped with just a single low-speed demonstration of the track, to rapidly learn a policy that can drive quickly and aggressively in the desired environment. This stage takes under 20 minutes, accelerated by pre-trained representations and enabled by a sample-efficient online RL procedure that effectively uses them.

The primary contribution of this work is FastRLAP, a system for autonomous learning of vision-based navigation that leverages diverse prior data and improves by practicing autonomously. We demonstrate our FastRLAP in challenging environments on a custom 1/10$^{\text{th}}$-scale RC car modified for real-world online RL. FastRLAP can autonomously practice and learn aggressive maneuvers using a novel autonomous practicing framework, improving by up to 40% over the demonstration lap and achieving performance close to a human expert. Notably, the online training phase typically takes less than 20 minutes (as little as 5 minutes!), depending on the size of the environment. During this time, the robot learns aggressive maneuvers, drifting, and maintaining a racing line, without any expert demonstrations. The training requires no human interventions and is fully autonomous. To the best of our knowledge, FastRLAP is the first instantiation of a vision-based mobile robotic system that uses model-free RL to autonomously practice high-speed driving maneuvers and improve online in the real world.

Algorithm 1: FastRLAP for Autonomous Practicing

Data: Prior navigation dataset $\mathcal{D}$, slow demo lap $\mathcal{B}_{\text{slow}}$
1 **Keys:** Pre-Training, Practicing, Online RL
2 **while** *Encoder is not converged* **do**
3 $s, a, s', \text{idx} \leftarrow \text{LoadData}(\mathcal{D})$
4 $g \leftarrow \text{LoadFutureData}(\mathcal{D}, \text{idx} + \text{RandomOffset}())$
5 $r \leftarrow \text{ComputeReward}(s, a, g)$
6 $\text{Train}_{\text{IQL}}((s, g), a, r, (s', g))$
7 **while** *True* **do**
8 **On Robot**
9 $s \leftarrow$Observe()
10 **if** s near g **then**
11 $g \leftarrow \text{NextCheckpoint}(g)$
12 $r \leftarrow \text{ComputeReward}(s_{\text{prev}}, a_{\text{prev}}, g)$
13 $\text{SendToWorkstation}(s_{\text{prev}}, a_{\text{prev}}, r, s, g)$
14 $a \sim \pi(\phi(s_{\text{image}}), s_{\text{proprio}}, g)$
15 Actuate(a)
16 **if** *Collision* or *Stuck* **then**
17 Execute recovery policy
18 **On Workstation**
19 $\text{ReceiveFromRobot}(\mathcal{B})$
20 $b \leftarrow \text{Sample}(\mathcal{B}), b_{\text{slow}} \leftarrow \text{Sample}(\mathcal{B}_{\text{slow}})$
21 $\pi, Q \leftarrow \text{Train}_{\text{RLPD}}(\pi, Q, b, b_{\text{slow}})$

II. Autonomous Practicing with RL

Our system for learning high-speed driving, FastRLAP, aims to enable autonomous practicing and sample-efficient end-to-end RL in the real world. FastRLAP has three components: a simple high-level finite state machine (FSM) for autonomous practicing, a pre-trained representation of visual observations, and a sample-efficient RL algorithm for online learning (see Fig. 1). The FSM (shown in blue) serves the dual purpose of selecting the next checkpoint for the online RL policy and automatically recovering from collisions, enabling autonomous practicing in the real world. The online RL policy (shown in orange) is trained *online* in the real world to reach goals commanded by the FSM, and continually improves to learn aggressive driving maneuvers. To provide for compute- and sample-efficient training of the online RL policy, we bootstrap it with an offline representation of navigation-specific visual features trained from prior data (shown in purple). For details of our system, see Appendix C.

A. Summary

Algorithm 1 summarizes the FastRLAP autonomous training framework. We use a diverse offline navigation dataset $\mathcal{D}$ to pre-train a navigation-relevant representation of visual observations using a goal-conditioned RL objective optimized with IQL (L6). We freeze the encoder trained with this process and and use it to encode visual observations for the actor and critic for the online learning phase (see Fig. 1, orange). The reward function for both the pre-training and online RL tasks is described by Eqn. 1.

During deployment in a previously unseen environment, the practicing FSM (blue) serves the dual purpose of commanding the next goal checkpoint to the low-level policy π (L11), and automatic collision recovery if the robot is stuck or in collision (L17). To enable fast training, the inference is split between the robot and a remote workstation with low-latency communication between them (see Sec. D for details).

The robot runs fast inference of the trained policy (L14) at 10Hz, and sends batches of online experience to the workstation (L13) to asynchronously update the actor and critic networks using RLPD (L21) as quickly as possible. This process enables FastRLAP to learn aggressive driving behavior from as little as 10-20 minutes of online experience.

III. Faster Lap Times with FastRLAP

In this section we present an experimental evaluation of FastRLAP in a variety of real-world and simulated environments. We consider several metrics to analyze the peak performance, as well as cumulative metrics during practice. The time-to-first-lap (T2F) represents the time taken to complete the first lap, starting from scratch. We track the best lap time achieved during training as well as the median time of last five laps completed to capture the converged behavior. Additionally, we list the median collisions in the last five laps to capture safety. To contextualize our results, we provide timing for laps driven in each environment by human drivers watching the robot from a third-person view ("Human Expert"), as well as the duration of the "slow demo" lap.

Env.	T2F (s)	Lap Times (s) Best	Median	Demo	Expert†	# Collisions Median
Real-A	114	32.7	39.0	54	25	0
Real-B	225	44.2	65.7	70	43	3
Real-C	117	10.9	11.7	17	7	0
Sim-D	925	104.1	107.0	286	112	0
Sim-E	174	18.0	18.1	36	19	0

TABLE I: Summary of experiments: FastRLAP can consistently learn aggressive driving policies in environments of varying difficulty levels, improving over the *demo lap* by over 40% and achieving lap times within 5% of the expert. †Note that the expert has access to privileged third-person observations.

Note that we used the *same hyperparameters* (i.e., network architecture, learning rate, reward coefficients) for all experiments discussed here, both in the real-world and simulation. See the appendix for a full list of hyperparameter values (Appendix G) and details about the simulation experiment (Appendix F).

A. Real-World Deployment

We deploy FastRLAP in three *previously unseen* indoor environments to demonstrate autonomous practicing in tightly constrained spaces. Before the start of training, we manually drive the robot around the course for a slow lap to define the rough layout of the track. This lap is used in two ways: (i) to generate a sequence of sparse checkpoints $\{c_i\}_{i=1}^{n_c}$ for the practicing FSM described in Sec. C1, and (ii) to provide a slow-speed demonstration for off-policy online actor-critic updates as described in Sec. C2. We describe the environments below described below are visualized in Fig. 7 (top).

Real-A represents a large loop (70 meters in length) through the interior of a carpeted building with glass walls and many open corridors. The course is defined by a sequence of $n_c = 4$ checkpoints spaced roughly 15-20 meters apart.

Real-B is a significantly larger course (∼120 meters in length) with multiple obstacles, defined by $n_c = 4$. The floor of this environment is a tiled and has very low friction, frequently causing dynamical effects such as over-/under-steer during cornering.

Real-C is a small but challenging indoor race course with two tight "hairpin" turns, taken at nearly the maximum steering angle and a tight "chicane" (a right-left sequence). Mastering this environment requires the robot to discover fast "racing lines" that minimize unnecessary steering, and carrying a high speed through the turns. This course is designated by $n_c = 3$ checkpoints. We extensively compare FastRLAP to alternative baselines in this environment.

Table I and Fig. 7 summarize the performance of our system in these environments. FastRLAP is able to consistently improve over the low-speed demonstration lap in under 4 laps, and nearly match human performance in Real-B in 30 minutes of real-world practice, without any human interventions. As training progresses, the achieved lap times continue to decrease, with the path taken by the robot becoming more optimized as a secondary effect of optimizing speed Fig. 6.

	T2F (s)	Lap Times (s) Best	Median	# Collisions Median
Real-World (Real-C)				
State-Based	274	12.7	18.8	3
Offline RL [7]	∞	–	–	–
No Pre-Training	233	12.7	20.0	1
ImageNet Pre-Training				
No Demo Lap	239	16.0	62.6	12
FastRLAP (Ours)	**117**	**10.9**	**13.3**	**0**
Human FPV	–	11.1	14.4	2
Human Oracle†	–	7.3	8.8	0
Simulation (Sim-E)				
State-Based	222	18.9	26.2	0
No Demo Lap	665	19.6	22.2	0
No Pre-Training	375	**17.8**	18.4	1
No Pseudo-Resets	405	21.7	25.1	0
FastRLAP (Ours)	**173**	**18.0**	**18.1**	0
Human Oracle†	–	18.6	18.9	0

TABLE II: Comparing to baselines in real-world and simulated environments, FastRLAP achieves consistently lower time-to-first lap (T2F), best and median lap times, and minimum collisions. Notably, FastRLAP can outperform a system with access to privileged state estimates, and a human driving from first-person view (FPV). In both environments, FastRLAP achieves close-to-oracle† driving performance, i.e., an expert human driving with *privileged* third-person observations of the robot, shown in gray. Offline RL failed to complete a single lap in Real-C, likely due to its inability to adapt the learned policies on-the-fly in novel environments.

Emergent behaviors: We find that directly maximizing the reward for reaching the next checkpoint as quickly as possible (Eqn. 1) leads to emergent behaviors in our system, visualized in Fig. 2. The system learns the concept of a "racing line", finding a smooth path through the lap and maximizing its speed through tight corners and chicanes (a–b). This can be seen in the speed profile through a tight corner in Fig. 2(a), where the robot learns to carry its speed into the *apex*, then brakes sharply to turn and accelerates out of the corner, to minimize the driving duration. In Fig. 2(c) with a low-friction surface, the policy learns to over-steer slightly when turning, drifting into the corner to achieve fast rotation without braking during the turn (c). Please see the supplemental material for videos of our system practicing and driving aggressively.

B. Comparative Analysis

We compare the performance of FastRLAP against the baselines and ablations mentioned in two environments: Real-C and Sim-E. we compare our approach to several baselines and ablations, and demonstrate the importance of each of the components of our method: pre-trained visual representations, online RL starting from a slow demo lap, and autonomous recovery behaviors to handle the reset-free environment. Specifically, we consider six baselines discussed in Appendix E.

In Real-C, our experiments (Tab. II) show that FastRLAP far outperforms the ablation without a demo lap in both time-to-first lap (T2F) and best lap time, while also encountering fewer collisions. Early in the training process, the demonstration helps the system make progress around

Fig. 2: Emergent behaviors with FastRLAP. (a, b) show the robot's telemetry data in environments `Sim-E` and `Real-B` after 30 minutes of autonomous practicing. We find that directly minimizing lap times with RL results in the emergent discovery of a "racing line", braking as late as possible to maintain speed in and out of tight corners (a, b) and drifting on slick surfaces (c). In a challenging outdoor environment `Sim-D`, the robot infers that the bridge is faster than driving through mud via visual correlation, learning a fast driving policy (d). *All the above graphics are rendered using telemetry from our real-world (b, c) and simulated (a, d) experiments. The third-person views are provided for visualization only, and are not available to our system.* Please see supplemental material for videos of these emergent behaviors.

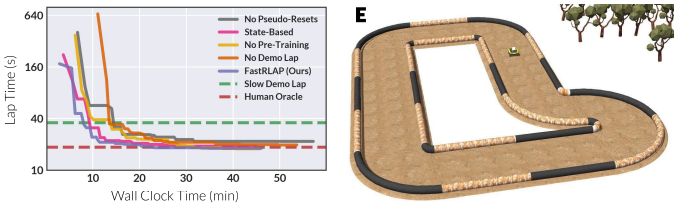

Fig. 3: Progression of running minimum lap times across different baselines in Tab. II in `Sim-E`. FastRLAP achieves expert-level performance in 20 minutes, learning efficiently from pixels as quickly as an agent with privileged state information, while achieving better overall performance by the end of training.

the course, enabling broad state coverage, which yields more useful exploration early on and eventually leads to better performance. The offline RL baseline completely failed to produce a usable policy, as it is unable to adapt online to new rich observations.

Several additional baselines were considered in the simulated `Sim-E` environment, summarized in Tab. II. The same trends hold, showing that the demo lap is very important to fast learning and achieving a low T2F. Similarly, removing pseudo-resets causes the system to get stuck for extended durations, resulting in a very slow first lap (even with a demonstration!). Fig. 3 shows the progression of the running minimum lap times for each method.

Analyzing the role of pre-training with offline RL, we find that FastRLAP initialized with a generic ImageNet encoder completes its first lap relatively quickly, achieving a T2F comparable to FastRLAP. However, its asymptotic performance is comparably poor: its best lap time across training in `Real-C` is only slightly better than the slow-lap demonstration. This suggests that while extracting general-purpose visual features (e.g., edges and gradients) may be sufficient for low-speed navigation, high-speed navigation requires consideration of task-specific features, such as depth or obstacle detection, that are better learned with task-specific pre-training.

Most surprisingly, learning directly from visual observations outperformed the variation with access to privileged state information in both simulated and real environments. This suggests that the features learned by the pre-trained encoder are *more informative* than simple localization estimates, because they generalize better: an obstacle looks the same across different positions and environments, while the state-based agent must learn a free-space representation of the entire environment inside its critic function via trial and error, bumping into each object before it can record its existence.

IV. Discussion

We presented a system for learning high-speed driving with reinforcement learning from rich observations, practicing autonomously in the real world. Our approach uses representations from prior data to initialize the policy, followed by sample-efficient online RL paired with a checkpoint-based navigation strategy to recover autonomously from collisions and continue practicing. Although deep RL is often believed to be inefficient and difficult to use in the real world, we demonstrate that with appropriate pre-training and several important design decisions, our system can actually learn effective driving strategies in under 20 minutes of real-world training. This result may seem quite surprising when viewed in contrast to prior work that uses simulated data [8], or hundreds of hours of training [9], and we believe it provides strong validation that deep RL can indeed be a viable tool for learning real-world policies even from raw images, when combined with appropriate pre-training and implemented in the context of an autonomous training framework.

A qualitative investigation of the policies learned by our system also reveals interesting emergent behavior. Although we bootstrap training with prior data (in other domains and from other robots) and a single (slow) demonstration lap, the learned policies exhibit skills like drifting and maintaining a racing line which deviate significantly from the behaviors seen in the prior data. Thus, the online RL process not only serves to robustify previously seen behavior, as observed in prior work incorporating offline data into real-world RL [10], but actually acquires new emergent behaviors building on the foundation established by the prior data. At the same time, our ablation experiments establish the importance of task-relevant pre-training, supporting the notion that representations learned from diverse robot navigation data serves as an effective foundation for downstream skill learning in much the same way that pre-training with self-supervised objectives enables efficient acquisition of downstream tasks in vision and NLP [11, 12].

References

[1] M. Bojarski *et al.*, "End to end learning for self-driving cars," 2016. 1

[2] M. Bansal, A. Krizhevsky, and A. Ogale, "ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst," in *Robotics: Science and Systems*, 2019. 1

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arxiv*, 2013. 1, 8

[4] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, 2020.

[5] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[6] O. Kroemer, S. Niekum, and G. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," *JMLR*, 2021. 1

[7] D. Shah, A. Bhorkar, H. Leen, I. Kostrikov, N. Rhinehart, and S. Levine, "Offline reinforcement learning for visual navigation," in *Conf. on Robot Learning*, 2022. 3, 7

[8] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Transactions on Robotics*, 2020. 4, 7

[9] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames," in *Intl. Conf. on Learning Representations (ICLR)*, 2020. 4

[10] A. Kumar, A. Singh, F. Ebert, Y. Yang, C. Finn, and S. Levine, "Pre-Training for Robots: Offline RL Enables Learning New Tasks from a Handful of Trials," *arXiv*, 2022. 4, 7

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv*, 2018. 4

[12] X. Chen, H. Fan, R. Girshick, and K. He, "Improved baselines with momentum contrastive learning," *arxiv*, 2020. 4

[13] S. Ross, G. Gordon, and D. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," in *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. 7

[14] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," in *AAAI Conference on Artificial Intelligence*, 2021. 7

[15] S. Thrun and T. M. Mitchell, "Lifelong robot learning," *Robotics and Autonomous Systems*, 1995. 7

[16] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *26th Annual International Conference on Machine Learning*, 2009.

[17] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in *International Conference on Machine Learning*, 2018.

[18] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman, "Teacher–student curriculum learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3732–3740, 2020.

[19] S. Sukhbaatar, Z. Lin, I. Kostrikov, G. Synnaeve, A. Szlam, and R. Fergus, "Intrinsic motivation and automatic curricula via asymmetric self-play," in *Intl. Conf. on Learning Representations (ICLR)*, 2018. 7

[20] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems," 2020. 7

[21] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv*, 2020. 7

[22] A. Villaflor, J. Dolan, and J. Schneider, "Fine-tuning offline reinforcement learning with model-based policy optimization," 2021.

[23] T. Xie, N. Jiang, H. Wang, C. Xiong, and Y. Bai, "Policy finetuning: Bridging sample-efficient offline and online reinforcement learning," in *Neural Information Processing Systems*, 2021.

[24] S. Lee, Y. Seo, K. Lee, P. Abbeel, and J. Shin, "Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble," in *Conf. on Robot Learning*, 2021. 7

[25] H. R. Walke, J. H. Yang, A. Yu, A. Kumar, J. Orbik, A. Singh, and S. Levine, "Don't start from scratch: Leveraging prior data to automate robotic reinforcement learning," in *OpenReview*, 2022. 7

[26] N. Gürtler, S. Blaes, P. Kolev, F. Widmaier, M. Wuthrich, S. Bauer, B. Schölkopf, and G. Martius, "Benchmarking offline reinforcement learning on real-robot hardware," in *Intl. Conf. on Learning Representations (ICLR)*, 2023. 7

[27] J. Funke, P. Theodosis, R. Hindiyeh, G. Stanek, K. Kritatakirana, C. Gerdes, D. Langer, M. Hernandez, B. Müller-Bessler, and B. Huhnke, "Up to the limits: Autonomous audi tts," in *IEEE Intelligent Vehicles Symposium*, 2012. 7

[28] N. Keivan and G. Sibley, "Realtime simulation-in-the-loop control for agile ground vehicles," in *Towards Autonomous Robotic Systems*, 2013.

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

[30] U. Rosolia and F. Borrelli, "Learning How to Autonomously Race a Car: A Predictive Control Approach," *IEEE Trans. on Control Systems Technology*, 2020. 7

[31] P. Drews, G. Williams, B. Goldfain, E. A. Theodorou, and J. M. Rehg, "Aggressive deep driving: Combining

convolutional neural networks and model predictive control," in *Conf. on Robot Learning*, 2017. 7

[32] ——, "Vision-based high-speed driving with a deep dynamic observer," *IEEE Robotics and Automation Letters*, 2019. 7

[33] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. A. Theodorou, and B. Boots, "Imitation learning for agile autonomous driving," *The International Journal of Robotics Research*, 2020. 7

[34] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, 2021. 7

[35] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Dürr, "Super-human performance in gran turismo sport using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4257–4264, 2021.

[36] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot, "Navigating to objects in the real world," *ArXiv*, vol. abs/2212.00922, 2022. 7

[37] M. Chang, A. Gupta, and S. Gupta, "Semantic visual navigation by watching youtube videos," in *Advances in Neural Information Processing Systems*, 2020. 7

[38] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J. Allen, V. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," *CoRR*, 2018. 7

[39] G. Kahn, P. Abbeel, and S. Levine, "Land: Learning to navigate from disengagements," *IEEE Robotics and Automation Letters*, 2021. 7

[40] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, "GNM: A General Navigation Model to Drive Any Robot," in *arXiV*, 2022. 7

[41] W. Han, S. Levine, and P. Abbeel, "Learning compound multi-step controllers under unknown dynamics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015. 7

[42] C. Richter and N. Roy, "Safe visual navigation via deep learning and novelty detection," in *Robotics: Science and Systems*, 2017.

[43] B. Eysenbach, S. Gu, J. Ibarz, and S. Levine, "Leave no Trace: Learning to Reset for Safe and Autonomous Reinforcement Learning," in *Intl. Conf. on Learning Representations (ICLR)*, 2018. 8

[44] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine, "The Ingredients of Real World Robotic Reinforcement Learning," in *Intl. Conf. on Learning Representations (ICLR)*, 2020. 8

[45] K. Lu, A. Grover, P. Abbeel, and I. Mordatch, "Reset-free lifelong learning with skill-space planning," in *Intl. Conf. on Learning Representations (ICLR)*, 2021.

[46] A. Sharma, K. Xu, N. Sardana, A. Gupta, K. Hausman, S. Levine, and C. Finn, "Autonomous reinforcement learning: Formalism and benchmarking," in *Intl. Conf. on Learning Representations (ICLR)*, 2022. 7

[47] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine, "Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021. 7

[48] S. Ha, P. Xu, Z. Tan, S. Levine, and J. Tan, "Learning to walk in the real world with minimal human effort," in *Conference on Robot Learning*, 2020. 7

[49] C. Sun, J. Orbik, C. M. Devin, B. H. Yang, A. Gupta, G. Berseth, and S. Levine, "Fully autonomous real-world reinforcement learning with applications to mobile manipulation," in *Conf. on Robot Learning*, 2022. 7

[50] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International Conference on Machine Learning*, 2019. 8

[51] Anonymous, "Efficient online reinforcement learning with offline data." OpenReview, 2023. 8

[52] X. Chen, C. Wang, Z. Zhou, and K. Ross, "Randomized Ensembled Double Q-Learning: Learning Fast Without a Model," Mar. 2021, arXiv:2101.05982 [cs]. 8

[53] D. Shah, B. Eysenbach, N. Rhinehart, and S. Levine, "Rapid exploration for open-world navigation with latent goal models," 2021. 8

[54] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arxiv*, 2021. 9

[55] "MIT RACECAR," 2014. [Online]. Available: https://racecar.mit.edu 9

[56] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1TENTH: An Open-source Evaluation Environment for Continuous Control and Reinforcement Learning," in *NeurIPS 2019 Competition and Demonstration Track*, 2020. 9

[57] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax 9

[58] I. Kostrikov, D. Yarats, and R. Fergus, "Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels," Mar. 2021. 9

[59] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 9

[60] S. Parisi, A. Rajeswaran, S. Purushwalkam, and A. Gupta, "The unsurprising effectiveness of pre-trained vision models for control," in *International Conference on Machine Learning*, 2022. 9

[61] C. Robotics, "Clearpath additional simulation worlds," 2022. [Online]. Available: https://github.com/clearpathrobotics/cpr_gazebo 10

APPENDIX

A. Related Work

Leveraging prior data to bootstrap online learning has been widely studied in the context of supervised learning [13], representation learning [14], continual learning [15–19], and RL [20]. Offline RL has proven particularly powerful due to its ability to learn actionable representations directly from existing large datasets, and some works have studied how it can be combined with fine-tuning through online interaction [21–24]. This has enabled a variety of robotic systems that can leverage a combination of offline data and online interaction to perform real-world tasks [10, 25, 26]. However, most such experiments focus on robotic manipulation or other scenarios that can evaluated in controlled workspaces. Instead, FastRLAP aims to perform fully autonomous practicing in unstructured environments, spanning over 120 meters. To address such real-world domains, FastRLAP incorporates pre-training data from a variety of environments and different robots, and employs a combination of diverse but less relevant demonstrations to enable effective bootstrapping. Additionally, FastRLAP integrates RL with a high-level practicing pipeline to enable greater autonomy during training.

Learning high-speed navigation has been approached in various ways. Typically, these systems rely either on highly accurate position information to define states [27–30], localize visual observations relative to a high-fidelity *global map* [31, 32], or operate via behavior cloning against some privileged expert which itself can access ground-truth state and mapping [33]. This can be prohibitive in unstructured environments, where (i) onboard state estimates can be highly inaccurate due to poor localization via noisy odometry or GPS measurements, and (ii) generating a high-fidelity map can be difficult or impossible. In contrast, FastRLAP learns aggressive driving behavior directly from vision, using only a coarse sequence of checkpoints, and can improve its behavior by self-practice without using privileged state information.

Prior successes in learning visual navigation for ground and aerial robots often involves either learning from large-scale simulated data [8, 34–36], passive data [37], human interventions [38, 39] or leveraging real-world data from other robots [40]. However, simulating off-road environments can be extremely challenging due to complex relationships between perception, vehicular dynamics, and terrain (Fig. 4), human interventions are time-consuming and expensive, and aggressive driving maneuvers tend to be closely adapted to the specific environments, calling for learning directly from *on-task* data. The closest prior work [7] uses offline RL for off-road driving, but does not support mechanisms to practice autonomously or adapt its behavior on-the-fly. We present the first navigation system that combines offline pre-training with fast, online RL training that can improve with experience and adapt to individual real-world environments autonomously.

A number of prior papers have studied the problem of autonomous real-world RL, often through the lens of safety or reset-free training, where the need for human interventions during training is minimized [41–46], and has been applied to robotic manipulation [25, 47], quadruped locomotion [48], and mobile manipulation [49]. We draw inspiration from these ideas to build an aggressive navigation system that uses a finite state machine to *practice* driving around a circuit. FastRLAP scales to novel environments, driving around 100+ meter courses, and can continually improve its performance.

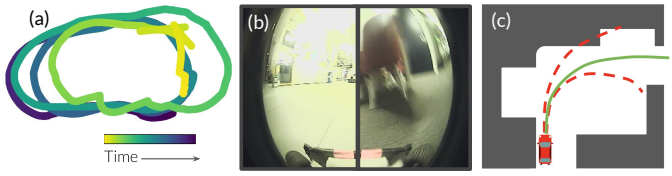

Fig. 4: High-speed visual navigation faces challenges due to sensing, perception, and dynamics: (a) noisy odometry and localization errors across multiple laps, (b) overexposure and motion blur in the visual observations, and (c) over-/under-steer due to complex dynamics.

B. Problem Formulation

The objective of our high-speed visual navigation task is to drive through a race course, defined as a sequence of position *checkpoints* $\{c_i\}$, in the minimum possible time. We frame this task as a Markov decision process (MDP) $\mathcal{M}(\mathcal{S}, \mathcal{A}, p, r)$, where $\mathcal{S}$ denotes the state space comprising of states $s = (V, v, \omega, \alpha, g, a_{\text{prev}})$. Here, $V \in \mathbb{R}^{128\times128\times3\times3}$ is a stacked sequence of the last 3 RGB images; $v, \omega, \alpha \in \mathbb{R}^3$ denote the linear velocity, angular velocity, and linear acceleration; the goal g is provided as a relative vector to the next checkpoint, written as a 2D unit vector and a distance; a_{prev} is the previous action. All measurements are specified relative to the robot's internal reference frame: the policy does not require information in any fixed external frame of reference. The action space $\mathcal{A}$ is specified by motor velocity targets corresponding to throttle and steering actions. Note that we do not allow the throttle command to be negative, the policy can only drive the robot forwards. p denotes the unknown transition dynamics, and r is a reward function corresponding to reaching c_i quickly (described in Sec. C2).

To make training in the real world practical and automated, we formulate this problem in the context of autonomous RL [46], where the robot is not provided with periodic resets or interventions when it collides with an obstacle or gets stuck, and needs to automatically recover on its own. The robot must *practice* driving around the lap to fully master the task and improve its performance over time *without* any human interventions. We measure the robot's performance along three metrics: (i) time to complete the first lap, (ii) final and median lap times over the course of training, and (iii) the mean number of collisions per lap over the course of training.

C. Autonomous Practicing with RL

1) Autonomous Practicing and Goal Checkpoint Selection: In the autonomous learning setting, the robot is expected to learn in the environment without any episodic "resets" or human interventions. In early stages of training, the RL policy

may make mistakes that lead to irrecoverable states, such as collisions. Without the help of a reset, the robot would get stuck forever and the learning algorithm may become degenerate due to collapse in the state distribution [44]. To overcome this, we use a simple FSM that switches between collision recovery and commanding a relevant goal checkpoint to the RL agent.

This FSM serves a dual purpose. When the RL policy reaches a goal checkpoint, as measured by a low-fidelity localization estimate (e.g., from visual-inertial odometry or GPS), the FSM commands a new goal corresponding to the next checkpoint in the course sequence $\{c_i\}$. This forces the learner to practice reaching all of the checkpoint goals on the race course sequentially. If the RL policy lands itself into an irrecoverable state (e.g., experiences a collision or becomes stuck, see Sec. D for implementation details), the FSM commands an automatic recovery policy to rescue the robot and provides a "pseudo-reset". We use a very simple scripted recovery policy to perturb the robot's state, which selects a random steering angle and drives backward for a short distance. Although other approaches such as using exploratory policies [44] or learning a recovery policy [43] have also been studied, we found this very simple strategy to be sufficient to allow the online RL procedure to learn directly in the real world without human interventions.

2) Online RL Training: The objective of the low-level policy π is to reach the goal checkpoints commanded by the FSM in the minimum possible time without colliding or getting stuck. Given the visual observations and a goal vector $\vec{g}$ from the FSM, π must parse the high-dimensional observations to understand its contents and plan a high-speed trajectory through it without colliding with obstacles or getting stuck. It is important to note that the goal checkpoints c_i are typically beyond line-of-sight (e.g., Fig. 1, blue), up to 40 meters away, and navigating between them requires the robot to learn a representation of the environment layout, correlating visual observations with possible *racing lines* that the robot could take to maintain high speed.

We design a simple reward function r that prioritizes maintaining maximum instantaneous velocity towards the next goal checkpoint, while also avoiding irrecoverable states that require the FSM to trigger the recovery policy, slowing down the robot's total lap time. We primarily define the reward as *speed-made-good*: the component of the instantaneous velocity in the direction facing the next checkpoint. We additionally add a penalty for becoming stuck (defined as failing to move despite commanding non-zero throttle) and colliding:

$$r(s,a) = \vec{v} \cdot \frac{\vec{g}}{\|\vec{g}\|} - C_{\text{stuck}} \mathbb{1}_{\text{stuck}} - C_{\text{collide}} \mathbb{1}_{\|a\|>A} \|a\|. \quad (1)$$

Here, $\vec{v}$ and $\vec{g}$ are the observed velocity and relative goal coordinates (included in the state observation s), a is lateral acceleration, $\mathbb{1}$ is the indicator function to detect irrecoverable states, and $C_{\text{stuck}}, C_{\text{collide}}, A > 0$ are constants.

To maximize the above reward and continually improve the robot's lap times, the system must learn from interactions with the environment in practice laps, using a *batch* of new interactions to update its learned behavior using off-policy RL [3, 50]. Such approaches benefit greatly by performing multiple training steps for each environment step, known as the update-to-data (UTD) ratio: a large UTD leads to efficient learning, but often suffers from overfitting. To overcome this, we train our policies with RLPD [51], a data-efficient off-policy RL algorithm that trains an ensemble of critics to avoid catastrophic overestimation [52] and can learn quickly using a combination of online interactions and a small amount of suboptimal, *on-task* data.

We obtain this *on-task* data by collecting a single *slow* lap in the target environment, similar to how a racing driver might perform a reconnaissance lap at low speed to familiarize themselves with the course before attempting laps at high speeds. While this data is very limited (under a minute in most environments) and does not contain fast-driving behaviors, prior work has shown that it can significantly accelerate online learning by stabilizing the critic from collapsing in early stages of training [51]. During the online training, we sample 50% of each training batch from this low-speed data, interleaved with 50% of data collected online. We found this to be critical to the efficiency of our system in our evaluations (Sec. F).

However, our system aims to learn effective high-speed navigation skills in as little as 10-20 minutes. At such low training times, the process is constrained not only by the robot's ability to collect data, but also by the *computational constraints* of training the neural network, so improvements in computationally efficiency actually translate directly into faster learning with higher UTD ratios. Therefore we combine the strengths of data-efficient online learning with a powerful pre-trained representation of visual observations to enable computation-efficient training.

3) Representation Learning with Offline RL: When training image-to-action policies, end-to-end RL allows gradients from the control objective to optimize the encoder. This results in a task-specific encoder that produces features that are most relevant to the agent's task, rather than general features (e.g., features necessary for classification or video prediction). Unfortunately, training directly on full images is very computationally expensive and unacceptably reduces the UTD ratio. Ideally, we would prefer to pre-train some encoder to produce task-relevant features *without* requiring new environment interactions, and then freeze the encoder during online training.

We address this by training the encoder with offline RL on an existing large-scale dataset with a similar (but not identical) objective. In particular, we use RECON [53], a large-scale navigation dataset collected by manually driving a Clearpath Jackal UGV outdoors at low-speeds. This dataset contains navigation trajectories from many environments and an entirely different robot, which importantly does *not* include aggressive high-speed driving. Thus, the role of pre-training is not to teach the robot how to drive quickly and efficiently, but only to extract a navigation-relevant representation to simplify the online learning problem. The aggressive, high-

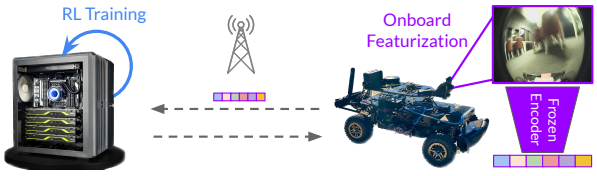

Fig. 5: For fast training, we featurize the visual observations using a pre-trained encoder onboard the robot (right) and run RL training on these embeddings on a wirelessly connected workstation (left).

speed driving behaviors necessary to solve the desired task must be learned through practice in the real world, building on this pre-trained foundation.

We apply goal-conditioned offline RL by selecting a 1:1 mixture of random goals and goals from the robot's future trajectory in this dataset, and use Implicit Q-Learning [54] to train a critic network (illustrated in Fig. 1, purple). We then take the learned encoder, which now encodes features relevant to the navigation task, and freeze it for training the aggressive driving policy and critic (orange) as illustrated in Sec. C2.

D. System Design for Online Learning

We instantiate our autonomous practicing system on a bespoke 1/10$^{\text{th}}$-scale autonomous rally car for high-speed navigation. While the base chassis was inspired by similar platforms [55, 56], the task of autonomous practicing with online learning adds a number of challenges that necessitate hardware and networking modifications. We describe these differences on top of a commercially available Traxxas Slash 4×4 Ultimate RC car (Fig. 5).

Sensing: Since our high-speed system operates directly on visual observations, we use a forward-facing PCB camera with a fisheye lens to obtain a low-latency stream of 128×128 RGB images with minimal motion blur. Since the RL policy requires coarse relative position estimates to intermediate checkpoints/subgoals, our system uses a low-fidelity state estimator. However, relying on wheel speeds and onboard IMU for local odometry is impractical due to wheel slippage. For this, we use a RealSense T265 tracking camera to provide local visual-inertial odometry estimates for the positions of the robot and intermediate checkpoints. In indoor environments, we orient the T265 to face upwards to minimize localization issues faced when the car bumps into obstacles.

Compute: Following F1TENTH [56], we use an NVIDIA Jetson Xavier NX for onboard compute. We process visual observations onboard using a pre-trained encoder (Sec. C3), and offload the RL training to a desktop workstation over a WiFi network or LTE (see Fig. 5). This has the dual benefits of low communication latency (since we only communicate low-dimensional features, under 100kB/s) and fast training (since only a subset of the layers need to be updated). To achieve a high UTD, we implement our training algorithm in JAX [57]. Using the just-in-time compiler to combine many update steps into a single optimized XLA function, we are able to increase our update rates to ∼800 actor updates per second. When combined with the on-robot featurization, this represents a ∼10x increase in throughput compared to training without these optimizations.

Actuation: The Markov assumption requires that the transition dynamics p only depend on the current state and action. However, typical "sensorless" motors used in RC cars exhibit *cogging* — a stochastic stuttering behavior that depends heavily on unobservable variables such as rotor positions — thus violating the assumption. This particularly manifests when steering sharply or starting from a standstill, actions that the RL agent must often make in exploration. We upgrade the system to use "sensored" motors to provide closed-loop startup sequencing without cogging. While the mechanical top speed of our system is nearly 30m/s, we cap the operational speed in indoor environments to 3.5m/s for safety.

Action continuity: The closed loop PID and servo actuators tend to act as a low-pass filter on their targets, smoothing out high-frequency changes. When sampling uncorrelated actions from an output distribution, this will result in a low signal-to-noise ratio between the commanded target actions and measured velocity observations, making critic learning via temporal difference learning difficult. To overcome this, we enforce *continuity* in the policy's outputs by constraining them to be near the previous action by modifying the action space: (i) instead of the standard $\tanh$ activation to limit the action space in $[-1, 1]$ (used by actor), we use a *shifted* $\tanh$ that limits it to the range $[a_{\text{prev}} - \delta, a_{\text{prev}} + \delta]$, i.e., near previous action, bounded by $\delta > 0$, and (ii) we append the previous action to the observed state. Please see the supplemental material for further details.

Detecting blocked states: Our autonomous practicing system uses a recovery policy when the robot fails to make forward progress (e.g., collision or stuck). We detect collisions via a simple heuristic condition: at each timestep, we compare the lateral acceleration of the robot to a threshold value; if that threshold is violated, the robot is in collision and a penalty is applied according to Eqn. 1. The "stuck" condition is detected using local odometry: if the robot has not moved by at 0.5m in the past three seconds, it is considered to be stuck and a pseudo-reset is performed.

E. Baselines

Offline RL: Ablating the online learning aspect, this baseline uses a policy trained purely offline, with access to 15 minutes of *expert* data. Note that this is *more* on-task offline data than is available to FastRLAP.

No Demo Lap: Ablating the online learning aspect, this baseline deploys FastRLAP *without* sampling any data from the low-speed demo lap $\mathcal{B}_{\text{slow}}$ (Alg. 1, L20).

No Pre-Training: Ablating offline pre-training, this baseline uses networks initialized from scratch and trained online using DrQ [58], a state-of-the-art pixel-based RL algorithm.

ImageNet Pre-Training: Ablating task-specific pre-training, this baseline uses the same encoder structure, but trained instead for image classification on ImageNet [59, 60] for extracting visual features.

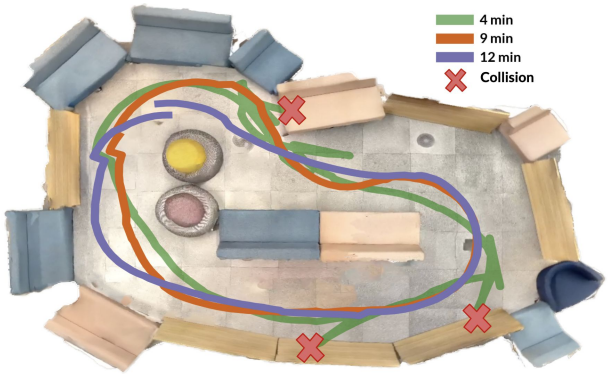

Fig. 6: Sample trajectories of FastRLAP practicing in a real-world indoor environment. Starting from a pre-trained representation, FastRLAP often collides into obstacles (green) and learns from it's mistakes to learn collision-free navigation (orange). By directly minimizing lap times, FastRLAP discovers a smooth racing line (purple), matching human driving performance. *Note that the 3D scan is shown for visualization only, and is not available to our system.*

No Pseudo-Resets: Ablating the FSM, this baseline deploys FastRLAP *without* the benefit of scripted pseudo-resets when it is stuck, requiring the robot to learn recovery behavior.

State-Based: This variation replaces visual observations with *privileged* state estimates in the form of an approximate 2D pose (measured by onboard visual-inertial odometry).

F. Simulated Experiments

We also test our approach in two simulated environments — a rally racing track, and a large, visually challenging off-road environment — to evaluate FastRLAP in more diverse settings. For these experiments we use a simulated Clearpath Jackal robot with identical observation space to our robot. Unlike our RC car-based platform, the Jackal uses a differential drive rather than an Ackermann steering setup; due to this difference the action space consists of linear and angular velocity targets rather than linear velocity and steering.

`Sim-D` (Fig. 7-D), is a large, complex world derived from Clearpath's simulation environments [61]. In this environment the robot must navigate around a large pool of mud, which shows non-binary traversability — greatly limiting the robot's dynamics and maximum speed — and should be avoided when possible. A bridge allows the robot to bypass the mud, but is narrow and can be identified visually. Our method successfully learns a high-performance policy in this environment, successfully correlating the mud's appearance with a low rewards, and selecting the optimal path after the first few laps of trial and error (see Fig. 2), achieving super-human lap times in under 10 laps (see Tab. I). This environment is particularly challenging since the mud can cause the robot to get irrecoverably stuck in a slow-speed zone; all alternative baselines and ablations failed to solve this task due to this reason. `Sim-E` is smaller-scale dirt track with sharp turns and chicanes, much like `Real-C`, making it a particularly interesting environment to study the emergence of *racing lines* and agile maneuvers.

G. Hyperparameters

All our experiments use the *same* set of hyperparameters and there is no environment-specific tuning used in the results presented in the paper. See Tab. III for a list of hyperparameters and Fig. 8 for a detailed network architecture.

Category	Hyperparameter	Value
Actor/Critic	Actor learning rate	3e-4
	Critic learning rate	3e-4
	Temperature learning rate	3e-4
	Actor network architecture	2x256
	Critic network architecture	2x256
	Initial target entropy	-3
	Entropy decay rate	1e-5
	Critic ensemble size	10
MDP/System	Discount factor	0.99
	Time step	0.1s
	Velocity target range (m/s)	$[0.5, 3.5]$
	Servo target range (rad)	$[-0.5, 0.5]$
	C_{collide} (real only)	$0.2s^2/m$
	C_{stuck}	-10
	Squashing range δ	0.2
Encoder	Layer count	4
	Convolution size	3x3
	Stride	2
	Hidden channels	32
IQL	Expectile	0.7
	Value network structure	same as critic

TABLE III: List of hyperparameters used throughout experiments

H. Additional Experimental Details

In addition to the metrics presented in the main paper, please see Tab. IV for a list of additional metrics per experiment, and Fig. 9 for lap time progression charts for each experiment. FastRLAP outperforms baselines in *all* environments, as measured by any suitable metric — time-to-first lap, best lap time, mean/median lap times, and minimum number of average and best-case collisions.

I. Implementation Details

The overall system was implemented using ROS 1 Noetic Ninjemys, with the inference and training code using JAX. We transfered tensors between the components of our system (new data from the robot to the workstation, and parameteres from the workstation to the robot) using ROS messages.

J. System Details

As mentioned in the paper, we squash the output of the actor dynamically to a range around the previous action a_{prev} to enforce continuity in output actions by constraining them to be no more than some positive constant δ away from the previous action in each dimension. We choose our activation function so that the interval $[a_{\text{prev}} - \delta, a_{\text{prev}} + \delta]$ is roughly mapped to itself. More precisely, we ensure that our activation $f_{a_{\text{prev}},\delta}(x)$ is accurate to first-order Taylor expansion around the

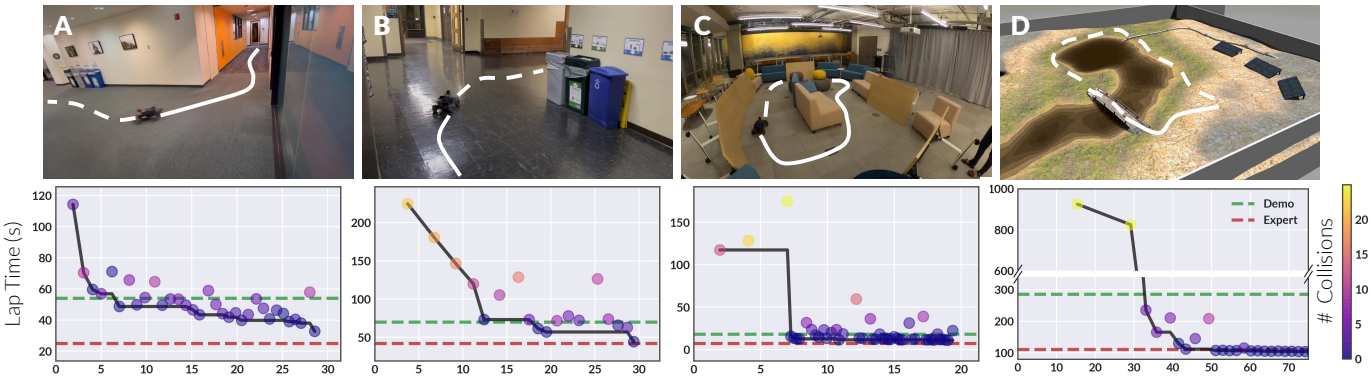

Fig. 7: Examples of high-speed driving in diverse, *previously unseen* environments (top) by autonomous practicing. FastRLAP improves its lap times (bottom, best-so-far shown) starting from a slow *demo lap* (green) and achieves close to expert lap times (red) in under 40 minutes.

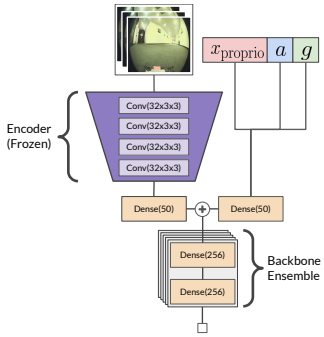

Fig. 8: Network architecture for the critic. Actor (and, for IQL, value) architectures are identical, but with only one backbone rather than an ensemble. Prorpioceptive information is concatenated with the action and the goal and fed through a dense layer, concatenated with the output of a convolutional encoder applied to a sequence of three camera images, and fed through a 2-layer MLP.

previous action. This yields the following activation function applied to the output of the actor:

$$f_{a_{\text{prev}},\delta}(x) = \tanh\left(\frac{(x - a_{\text{prev}})}{\delta}\right)\delta + a_{\text{prev}}$$

Figure 10 shows a graphical depiction of this activation function.

K. Simulation Environment

The Clearpath Jackal used for simulations differs from the real environments primarily in its action space, which (as a differential drive robot) allows turning in place. We limit the linear velocity actions of the robot to $[-1, 2]$ and the angular velocity actions to $[-1.0, 1.0]$. Simulated position measurement is provided in lieu of the RealSense tracker for determining relative goal locations.

L. Supplemental Video

Please see the attached video for an overview of our online practicing framework along with video clips of learned driving behavior in the real-world environments presented in the main paper.

M. Preliminary Code Release

Please see `fastrlap_code.tar.gz` for the training and robot-side inference code as well as modified simulation environments.

Course name	Method	Lap time (s) first	best	median	median (5)	# collisions best	total	mean	median	mean (5)	median (5)
Real-A	FastRLAP	114.31	32.74	49.51	38.99	0	82	2.48	2	2.00	0
Real-B	FastRLAP	224.70	44.21	73.71	65.72	0	135	7.50	7	4.20	3
Real-C	States	274.42	12.70	50.13	18.88	0	190	12.67	4	3.40	2
Real-C	Ours	117.38	10.90	13.27	11.69	0	126	2.74	0	0.00	0
Real-C	No slow lap	239.21	16.01	64.51	62.62	0	206	22.89	12	8.80	11
Real-C	Human Oracle	54.40	7.21	10.08	8.79	0	7	1.00	0	0.00	0
Real-C	ImageNet encoder	49.75	19.66	30.20	21.12	0	168	6.46	1	0.00	0
Real-C	No pretraining	232.79	12.70	21.60	19.99	0	174	9.67	1	1.80	1
Sim-D	FastRLAP	925.08	104.19	107.00	107.00	0	157	4.76	0	0.00	0
Sim-E	Blind	222.17	18.89	21.70	19.50	0	127	1.18	0	1.20	0
Sim-E	States	222.17	19.10	23.69	26.20	0	113	1.30	0	1.00	1
Sim-E	FastRLAP	174.30	17.99	19.10	18.10	0	48	0.42	0	0.00	0
Sim-E	No slow lap	665.04	19.70	23.33	22.20	0	90	0.95	0	0.00	0
Sim-E	No pretraining	375.26	17.80	21.90	18.39	0	100	1.14	0	0.00	0
Sim-E	No pseudo-resets	405.02	21.70	26.31	25.10	0	156	1.64	0	0.20	0

TABLE IV: Detailed statistics of all runs

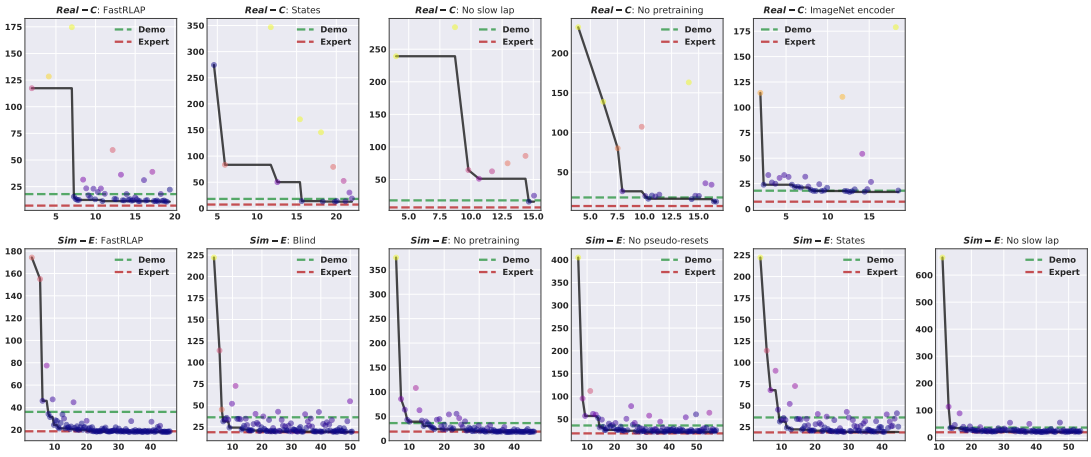

Fig. 9: Detailed laptime progression charts for all baselines

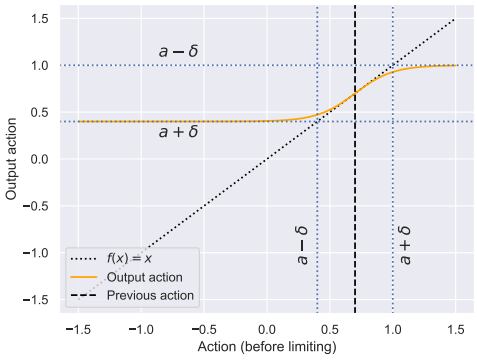

Fig. 10: Our activation function (actor output only) ensures action continuity.