A Super-human Vision-based Reinforcement Learning Agent for Autonomous Racing in Gran Turismo

Miguel Vasco[*†]
KTH Royal Institute of Technology
miguelsv@kth.se

Takuma Seno[*]
Sony AI
takuma.seno@sony.com

Kenta Kawamoto
Sony AI
kenta.kawamoto@sony.com

Kaushik Subramanian
Sony AI
kaushik.subramanian@sony.com

Peter R. Wurman
Sony AI
peter.wurman@sony.com

Peter Stone
Sony AI
The University of Texas at Austin
peter.stone@sony.com

Abstract

Racing autonomous cars faster than the best human drivers has been a longstanding grand challenge for the fields of Artificial Intelligence and robotics. Recently, an end-to-end deep reinforcement learning agent met this challenge in a high-fidelity racing simulator, Gran Turismo. However, this agent relied on global features that require instrumentation external to the car. This paper introduces, to the best of our knowledge, the first super-human car racing agent whose sensor input is purely local to the car, namely pixels from an ego-centric camera view and quantities that can be sensed from on-board the car, such as the car's velocity. By leveraging global features only at training time, the learned agent is able to outperform the best human drivers in time trial (one car on the track at a time) races using only local input features. The resulting agent is evaluated in Gran Turismo 7 on multiple tracks and cars. Detailed ablation experiments demonstrate the agent's strong reliance on visual inputs, making it the first vision-based super-human car racing agent.

1 Introduction

Autonomous car racing is a challenging task for intelligent artificial agents, where performance gaps in milliseconds can be the difference between winning and losing a race. To effectively perform this task, agents must be able to (i) process high-dimensional sensor data to estimate the state of the autonomous vehicle, (ii) continuously plan optimal driving lines while avoiding obstacles and other vehicles, and (iii) control the vehicle, while accounting for its nonlinear behavior and the conditions of the road (Betz et al., 2022). Recently, deep reinforcement learning (RL) methods have shown great promise in learning racing behavior through trial-and-error interaction with the race track environment, without the need for extensive domain knowledge (Jaritz et al., 2018; Imamura et al., 2021; Cai et al., 2021; Remonda et al., 2021; Herman et al., 2021). Despite their ability to consistently drive around the track, most learned policies still perform slower than median human racers (Cai et al., 2021; Herman et al., 2021).

In this work, we focus on learning RL (Sutton & Barto, 2018) agents that are able to achieve *super-human* performance in autonomous racing tasks, i.e., they are able to outperform (in terms of lap time) the best human drivers on a given track. Recently, two RL methods have reported super-human performance in Gran Turismo, a high-fidelity racing simulator (Fuchs et al., 2021;

[*]Equal contribution.
[†]Work done during his internship at Tokyo Laboratory, Sony AI.

Wurman et al., 2022). However, during execution these methods rely on *global features*, such as forward looking course shape information, that require instrumentation external to the vehicle. In contrast, human drivers rely on car-centric *local features* to race, such as *visual information* and *propriocentric features* that can be estimated from on-board the vehicle (e.g. velocity of the car). In this work, we ask the question: *can we train RL agents that are able to consistently achieve super-human performance when provided only with local features at execution time?*

Learning optimal racing behavior requires information that might not be possible to access only through local features at each time step: for example, in a tight corner the agent might be unable to see the apex and the end of the curve, which are fundamental to select an optimal driving trajectory. To overcome this challenge, and motivated by recent works on reinforcement learning with partial observability (Pinto et al., 2017; Salter et al., 2021; Sinha & Mahajan, 2023; Baisero et al., 2022), we leverage a distributed *asymmetric* actor-critic architecture that provides global features to the critic during training. The policy (actor) is provided only with local features, i.e., image and propriocentric features, allowing the agent to race without global information at execution time.

We evaluate our agent in Gran Turismo 7 (GT7), a high-fidelity driving simulator for PlayStation®. We show that our agent consistently achieves faster lap times than all human reference drivers (over 130K per scenario) across multiple time trial races, in which the goal is to complete a lap around the track in the minimum amount of time. Additionally, we conduct an extensive ablation study that shows the significant contribution of local features and of the asymmetric training scheme to the agent's overall performance. Furthermore, we perform a qualitative study on the learned policy and highlight novel driving lines, in comparison with the best human reference drivers, and demonstrate the strong reliance on image features for the agent's decision-making. To the best of our knowledge, we present the first vision-based super-human car racing agent.

In summary, our contributions are three-fold: (i) we contribute a vision-based RL agent for autonomous racing that employs an asymmetrical actor-critic training scheme; (ii) our agent consistently outperforms all human reference drivers (over 130K) across multiple time trial races in Gran Turismo 7, while having access only to local features, and performs on par with other super-human racing agents that rely on global features during execution; (iii) we conduct an extensive evaluation study that highlights the importance of the asymmetrical training scheme, novel driving behavior in comparison with the best human reference drivers, and the strong dependence on image features for the decision-making of our agent.

2 Related Work

Autonomous Racing (AR) is a subfield of autonomous driving research that concerns autonomous vehicles that operate at their dynamical and power limits within racing environments (such as racing circuits) (Betz et al., 2022). Research in autonomous racing can traditionally be categorized into perception (Massa et al., 2020; Peng et al., 2021), planning (Herrmann et al., 2020; Vázquez et al., 2020) and control (Williams et al., 2018; Hao et al., 2022). In our paper, instead, we employ end-to-end RL that combines perception, planning, and control into a single process, in particular with vision inputs, and is able to achieve super-human performance. For an extended version of the related work, including a discussion on asymmetrical training in RL, please refer to Appendix A.

Vision-based Reinforcement Learning for AR: RL has been shown to be a promising approach to learn competitive racing behavior (O'Kelly et al., 2019; Herman et al., 2021; Rong et al., 2020; Dosovitskiy et al., 2017). Jaritz et al. (2018) explore vision-based RL for driving agents in the context of a rally game. However, the authors find that their method does not achieve optimal racing trajectories, "lacking anticipation", and is unable to complete the racing tracks without colliding several times with obstacles. Cai et al. (2021) described an approach that merges imitation learning and model-based RL to learn racing behavior. However, their method requires expert-level demonstrations to pretrain the policy. The racing performance of current vision-based methods is still sub-optimal. Some works lack a performance comparison against humans (Remonda et al., 2021; Jaritz et al., 2018) or, when such comparison is made, the methods still under perform significantly

Figure 1: Our vision-based RL agent for autonomous car racing. (Left) We exploit an asymmetric actor-critic architecture to train our agent: the policy network π_ϕ is provided with propriocentric information $\mathbf{o}^p$ and image features $\mathbf{h}^i$, encoded with a convolutional neural network q_θ, to output actions $\mathbf{a}$. The critic network Q_ψ is provided with local propriocentric observations and global observations $\mathbf{o}^g$ (i.e., course shape information) to predict quantiles of future returns. (Right) During execution, our agent only receives local features from the Gran Turismo 7 simulator.

against median human users (Cai et al., 2021; Herman et al., 2021). Imamura et al. (2021) also explores vision-based RL for racing agents using a pretrained image encoder on random observations of the track environment. The authors report that they are unable to outperform the best human players. To the best of our knowledge, we contribute the first vision-based agent that is able to consistently outperform all human reference drivers across multiple time trial races.

Super-human Performance in AR: Recently, Fuchs et al. (2021) and Wurman et al. (2022) have reported super-human performance by autonomous racing RL agents in time-trial and actual racing, respectively. Fuchs et al. (2021) introduced a model-free RL approach and designed a novel proxy reward that considers the progress of the agent in the course. Their method is able to achieve super-human performance in time trial races in Gran Turismo Sport, a highly realistic racing simulation. Wurman et al. (2022) introduced Gran Turismo Sophy (GT Sophy), an RL agent that is able to achieve super-human performance both in time trial and racing scenarios with multiple opponents. To achieve super-human performance both approaches require global features (e.g., forward looking course shape information) at execution time.

3 Methodology

To train a vision-based autonomous racing RL agent that achieves super-human performance without global features at execution time, we design a distributed *asymmetric* actor-critic architecture and employ Quantile Regression Soft Actor-Critic (QR-SAC), a recently introduced distributional RL algorithm (Wurman et al., 2022). Our method is depicted in Figure 1.

3.1 Observation Space

We build the multimodal observations $\mathbf{o}$ of our racing agent at time step t, following,

$$\mathbf{o}_t = (\mathbf{o}_t^l, \mathbf{o}_t^g),$$

where $\mathbf{o}_t^l$ corresponds to local (to the car) features and $\mathbf{o}_t^g \in \mathbb{R}^{531}$ corresponds to the global features (i.e., course shape information). As local features $\mathbf{o}_t^l = (\mathbf{o}_t^i, \mathbf{o}_t^p)$ we consider an image $\mathbf{o}_t^i \in \mathbb{R}^{64\times64\times3}$ and propriocentric information $\mathbf{o}_t^p \in \mathbb{R}^{17}$.

Image features ($\mathbf{o}_t^i$): At each time step we extract an image directly from the game, considering a first-person view of the track ahead (a camera view denoted by `Normal view` in the game), from the front of the vehicle. The image is scaled from 1920×1080 (the native resolution of the game) to 64×64, with RGB information. Empirically we found this resolution to be sufficient to allow the

agent to race at super-human speeds, as we show in Section 5.1. Given the low resolution of the image observation, we turned off all extra information on the screen like the heads-up-display (HUD). As an artifact of the simulator, the image observation also includes the car's rear-view mirror. In Appendix C.3 we show that our agent can still consistently achieve super-human lap times without any rear-view mirror information. We provide examples of image observations in Figure 2 and in Appendix B.

Propriocentric features ($\mathbf{o}_t^p$): We select features for $\mathbf{o}_t^p$ that can be easily accessible in a real-world autonomous racing scenario,

$$\mathbf{o}_t^p = [\mathbf{v}_t, \dot{\mathbf{v}}_t, \mathbf{v}_t^r, \mathbf{c}_t, \mathbf{h}_t^a, \mathbf{h}_t^d],$$

where $\mathbf{v}_t \in \mathbb{R}^3$ corresponds to the linear velocity of the car in its local coordinate system, $\dot{\mathbf{v}}_t \in \mathbb{R}^3$ corresponds to the linear acceleration of the car, $\mathbf{v}_t^r \in \mathbb{R}^3$ corresponds to the angular velocity of the car, $\mathbf{c}_t \in \mathbb{R}^3$ corresponds to the current steering, throttle and brake vector, $\mathbf{h}_t^a \in \mathbb{R}^3$ corresponds to a short history of the steering angles in the last three steps and $\mathbf{h}_t^d \in \mathbb{R}^2$ corresponds to the delta steering changes in the last three steps. The velocity and acceleration features can be estimated using inertial measurement units (IMU), which are often included in real autonomous vehicles (Betz et al., 2022), and the steering features can be easily extracted from the car's guidance system.

Global features ($\mathbf{o}_t^g$): Following Wurman et al. (2022), we explore *course point information* as global features. Course points are built using the shape of the track, including track limits of the left and right, and a center line of the track. At each time step, the range of the course points is a function of the current velocity of the vehicle: we consider the 3D relative coordinates of the course points ahead of the agent from 0.1 sec up to 6 sec ahead (maintaining the current velocity), equally spaced on 0.1 sec intervals. This results in 59 course points for each course line (left, center and right). In Appendix C, we evaluate the effect of course point range on the performance of our agent.

3.2 Action Space

Similarly to Fuchs et al. (2021); Wurman et al. (2022), we define the actions of our agent $\mathbf{a}_t \in \mathbb{R}^2$, consisting of a *delta steering angle* and a *combined throttle and brake* value. The delta steering angle at a single time step is limited within $[-3°, +3°]$ to prevent steering changes from exceeding human limitations. The combined throttle and brake is represented by a normalized scalar in $[-1, +1]$. Values in the positive range represent throttle and ones in the negative range correspond to brake. The gear shift of the vehicle is controlled by automatic transmission. We set the control frequency to 10 Hz and the game, running at 60 Hz, linearly interpolates the steering angle between steps.

Due to technical constraints when retrieving images from the game in real-time, we utilized a synchronous communication process between the game and our agent, instead of asynchronous communication. This mode ensures alignment between the image and propriocentric features. In this mode our agent does not execute its policy in real-time during training due to the synchronicity of the simulator. However, we show in Appendix C.2 that executing the trained policies asynchronously, i.e., in real-time, still allows our agent to achieve super-human performance. In Appendix I, we provide more details regarding our communication configuration.

3.3 Reward Function

Following Wurman et al. (2022), we designed the reward function of the agent as the weighted combination of multiple components,

$$r_t = r_t^p + \lambda^o r_t^o + \lambda^w r_t^w + \lambda^s r_t^s + \lambda^h r_t^h.$$

Course progress (r^p): We formulate the lap time minimization problem as a course progress maximization problem: we compute the progress of the vehicle position, projected onto the center line of the track, since the last step;

Table 1: Time trial scenarios for the evaluation of our vision-based racing agent. We evaluate our approach across different tracks, cars and track conditions. We also consider different tire settings: racing soft (RS), sport soft (SS) and sport medium (SM). We compare our agent against over 130K human players in each scenario.

Scenario	Track	Condition	Car	Tire	Number of participants
`Monza`	Autodromo Nazionale Monza, Italy	Day, Clear	Ferrari 330 P4 '67	RS	138,306
`Tokyo`	Tokyo Expressway - Central Clockwise, Japan	Night, Clear	NISMO 400R '95	SS	131,598
`Spa`	Circuit de Spa-Francorchamps, Belgium	Day, Cloudy	Alfa Romeo 4C Launch Edition '14	SM	144,308

Off-course penalty (r^o): We define a shortcut penalty to prevent the agent from violating racing rules by cutting corners, $r_t^o = -(s_t^o - s_{t-1}^o)|\mathbf{v}_t|$, where s^o is the total time that the vehicle had (at least) three tires outside the track limits.

Wall penalty (r^w): We define a wall-hit penalty to prevent the agent from exploiting walls to quickly change its direction of movement, $r_t^w = -(s_t^w - s_{t-1}^w)|\mathbf{v}_t|$, where s^w is the total time the vehicle was in contact with a wall in the track.

Steering change penalty (r^s): To discourage large changes of steering angles in a single step, we define a steering change penalty, $r_t^s = -|\theta_t^s - \theta_{t-1}^s|$ where θ_t^s is the steering angle in radian at time step t;

Steering history penalty (r^h): We additionally define a steering history penalty to discourage the agent to make inconsistent decisions in a short period of time,

$$r_t^h = -m_t \cdot 1/(1 + \exp(-c^s \cdot (\Delta_t - c^o))),$$

where $\Delta_t = |\delta_t| + |\delta_{t-1}|$, $\delta_t = \theta_t^s - \theta_{t-1}^s$, $m_t = \mathbb{I}_{\delta_t > c^d} \cdot \mathbb{I}_{\delta_{t-1} > c^d} \cdot \mathbb{I}_{\text{sgn}(\delta_t) \neq \text{sgn}(\delta_{t-1})}$, c^d is a threshold angle, c^s is a sensitivity factor and c^o is an offset value. In Appendix J we provide the reward function parameter values used in our evaluation.

3.4 Training Architecture

We train our agent using QR-SAC, a distributional RL extension to Soft Actor-Critic (Haarnoja et al., 2018) with multi-step TD error. In QR-SAC, critic functions are represented with a quantile distribution function (Dabney et al., 2018) that estimates quantiles of returns.

To achieve super-human performance at execution time, inspired by recent works in RL under partial observability (Pinto et al., 2017; Salter et al., 2021; Sinha & Mahajan, 2023; Baisero et al., 2022), we consider an *asymmetric* actor-critic architecture for QR-SAC training, as shown in Figure 1. During training, the critic functions are provided with global features $\mathbf{o}^g$, instead of image observations $\mathbf{o}^i$, allowing them to learn accurate returns. The policy is only provided with image and propriocentric features $\mathbf{o}^p$. Since the policy does not depend on the course points to predict actions, the agent is able to race at execution time only with local observations. We detail our model architecture in Appendix I and our training hyperparameters in Appendix J.

4 Evaluation

We evaluate our agents in time trial tasks, where the goal is to complete a lap across the track in the minimum time possible. In this section we cover the track and car scenarios used for testing, the racing baselines, and the human player data used for comparisons.

Figure 2: Examples of 64×64 image observations in (left) `Monza`, (middle) `Tokyo`, and (right) `Spa`.

Method	Monza	Tokyo	Spa
Built-in AI	107.828 ±—	87.905 ±—	168.280 ±—
Fastest Human	104.378 ±—	80.782 ±—	157.796 ±—
GT Sophy (Wurman et al., 2022)	**104.281** ±0.061	**80.227** ±0.047	**157.424** ±0.038
Our Agent	**104.300** ±0.050	**80.401** ±0.091	**157.554** ±0.055

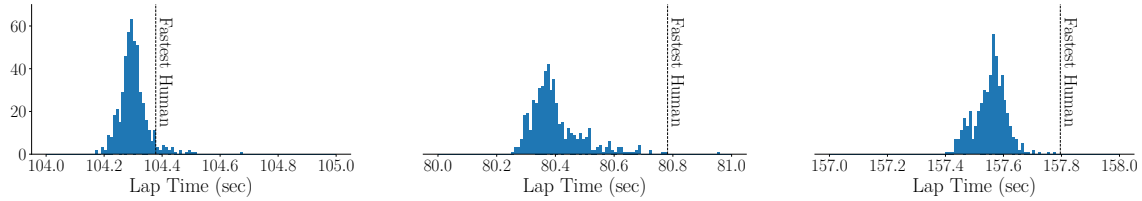


Figure 3: (Top) Lap time across all scenarios. We consider five randomly-seeded training runs and average the results over 500 evaluation laps, with 100 laps executed by the fastest policy in each training run. We highlight results that are significantly faster than the fastest human time (using a Wilcoxon signed-rank test, with $p < 0.001$); (Bottom) Distribution of lap times in `Monza` (left), `Tokyo` (middle) and `Spa` (right).

4.1 Scenarios

We evaluate our agent in three scenarios in GT7 with different combinations of cars, tracks, and conditions (track time and weather): `Monza`, `Tokyo`, and `Spa`, modeled after real-world circuits and roads. These scenarios were selected based on past GT7 online race events, where human players joined time trial races using the exact same car setup as our approach[1].We provide a more detailed description of the evaluation scenarios in Table 1 and image observations of the different scenarios in Figure 2 and in Appendix B.

4.2 Baselines

GT Sophy (Wurman et al., 2022): We use GT Sophy, a recently introduced super-human racing agent for Gran Turismo, as a baseline in our experiments. As this baseline was shown to be able to outperform the best human drivers and exploits global features to act, we consider its performance as an upper-bound to the performance of our method. We modify the action space of GT Sophy, which originally outputs absolute steering angles rather than delta angles, to match the action space of our agent. Moreover, we use the same training hyperparameters and reward function as our method. We train this baseline for GT7 using the same training method described in Wurman et al. (2022).

Human Players: Human player data was provided by *Polyphony Digital Inc.*, the development studio of Gran Turismo. For each scenario we collected over 130K lap times and trajectories. We consider our agent to have super-human performance if it is able to achieve a faster lap time than the one achieved by the fastest human player in each scenario.

Built-in AI: The built-in AI of GT7 follows a pre-defined human expert trajectory using a rule-based tracking approach, similar to MPC methods, and serves as a traditional control-based baseline. We report the minimum lap time of the built-in AI after executing 4 laps in each scenario.

[1]For more details regarding the online race events, refer to `https://www.gran-turismo.com/us/gt7/sportmode/`.

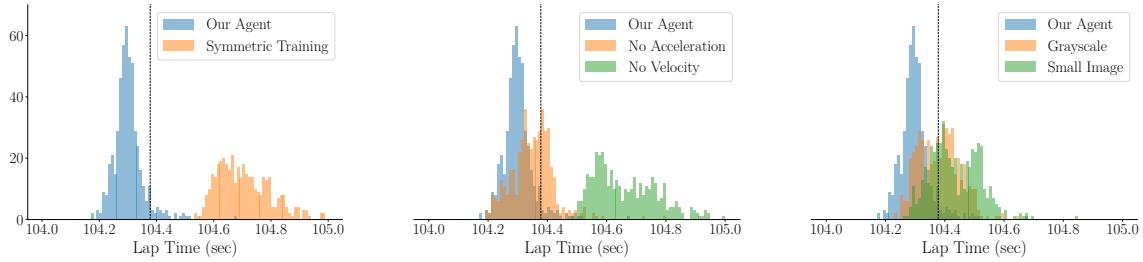


Figure 4: Performance study of our racing agent in the `Monza` scenario in relation to the training architecture (left), local features (middle) and the image feature (right). We consider five randomly-seeded training runs and show the distribution of 500 evaluation laps, with 100 laps executed by the fastest policy in each training run. We highlight the lap time of the fastest human player (black line). One symmetric run failed to learn meaningful behavior and we exclude it from the analysis.

5 Results

We show the minimum lap times achieved by the different agents in Figure 3. In all three scenarios, our agent achieves super-human performance, with lap times that significantly surpass the performance of the best human player. Our agent also achieves comparable performance to GT Sophy, despite not having any global features at execution time. Our agent achieves this level of performance *consistently*, with small variation across the randomly-seeded runs, as shown in the training curves in Appendix D, and across the different evaluation laps: for `Monza` we outperform the fastest human player in 94.0% of the laps, for `Tokyo` in 99.8% of the laps and in `Spa` in all the laps. We note that the distribution in lap times is a result of the high-fidelity physics engine of the simulator, where small numerical differences can result in different behaviors, thus preventing the agent from repeating the same trajectory across multiple laps.

5.1 Ablation Study

We define ablated versions of our method to evaluate the contribution of different architectural and training choices to the performance of our method, in particular regarding (i) the training architecture, (ii) local features, and (iii) the image feature. For (i) we employ a symmetric training scheme, where we replace the course points in the critic's input with image observations (*symmetric training*). For (ii) we remove acceleration features, (*no acceleration*) and velocity features (*no velocity*) from $\mathbf{o}^p$, and remove image features (*no image*) from $\mathbf{o}^l$. For (iii) we remove color from the image observation (*grayscale*) and reduce the size of the image observation to 32×32 (*small image*).

We present the results of our ablation study in Figure 4. Regarding (i) the results show that providing the critic with global features during training is fundamental for the performance of our agent as it mitigates the partial observability of the environment, allowing for a better estimate of the returns of the policy. Regarding (ii) we observe that removing velocity features results in a decrease in the performance of our agent as, naturally, velocity information is fundamental to racing at a consistently high level around the track. Regarding (iii), the results highlight that both color and a larger size of the image helps improve the performance of our method. Additionally, we found that the *no im-*

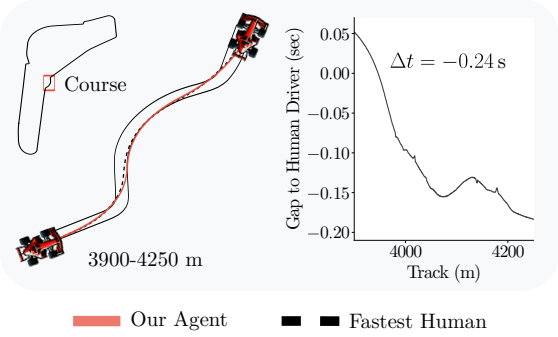


Figure 5: (Left) Trajectory comparison in the `Monza` track between our agent and the fastest human player in a chicane section. (Right) Gap of our agent to the human driver. Lower is better.

age agent is unable to drive. As such, we do not present this condition in Figure 4. This result further highlights the importance of visual information for our agent. In Appendix C we report on additional studies regarding the range of course points for the critic's input, revealing that we can further improve the performance of our agent by fine-tuning this parameter; and on the synchronous communication of our training pipeline, highlighting that we can execute our trained policies in an asynchronous (i.e., real-time) version of our simulator without significant loss in performance.

5.2 Qualitative Policy Study

We qualitatively evaluate the policy of our agent with regards to its trajectory against the best human drivers and the importance of visual features for the decision-making of our agent.

Trajectory Analysis: In Figure 5 we compare the trajectories of our agent and of the best human player in a chicane section. Our agent takes a driving line closer to the track limits, slowing down only 16.5% of what the human driver slows down in the section, thus gaining 0.24 seconds. The novel racing behavior exhibited by our agent motivates its use as a training tool for human drivers. We note that while the agent is able to achieve super-human lap times, it is not faster than the best human player across all segments of the track. We present an extended version of this study, including comparisons to GT Sophy, across all scenarios in Appendix E.

Visual Analysis: We employ Guided Gradient-weighted Class Activation Mapping (GGC) (Selvaraju et al., 2017), a visual analysis tool for image-based classification tasks (Arrieta et al., 2020; Linardatos et al., 2020), to understand what high-level features in the input image are relevant to the decision-making of our agent. We modify the original algorithm for RL tasks, as described in Appendix G. In Figure 6, we present GGC visualizations for the steering action of our agent in the `Monza` track.

The results show a distinct pattern of behavior for different sections in the track. In long straights, far-away visual features, such as horizon of the track or the tree line, are more significant for the policy of our agent than close visual features, such as the curb of the track. Naturally, in these sections, the agent is travelling at high-speeds and mostly needs to focus on identifying where the straight ends. However, in chicanes and tight curves, our agent focuses on the closer curbs of the track which are fundamental to successfully change its direction without going off-track and incurring on a penalty. This *gaze-like* behaviour echoes the one exhibited by human drivers (Rito Lima et al., 2020; Van Leeuwen et al., 2017): during straight segments, the human eye gaze focuses straight ahead, with a stable distance in the horizon, and during curves the eye gaze is focused on the inner tangent (apex) of the curve. Additionally, our agent uses the uniqueness of the visual features in the track to localize itself: we see that it considers both forward features (track limits and horizon) and backward features (rear-view mirror) in its decision-making. We consider that the focus on the rear-view mirror indicates that the trained policy exploits the static track layout for localization. We provide additional visualizations for all scenarios in Appendix H.

5.3 Perturbation Study

We conduct an extensive evaluation of the robustness of our agent to input perturbations and different track/car conditions in Appendix F. Amongst other results, the study highlights the importance of visual information for our agent: (i) changes in the lighting conditions of the environment (due to racing at a different time of the day) degrade significantly the performance of our agent; (ii) adversarial perturbations to the image observation, in particular to the image features computed using GGC, also degrade significantly the performance of our agent.

6 Conclusions

In this paper we presented the first super-human, vision-based reinforcement learning agent for autonomous car racing. To achieve this level of performance, we leverage an asymmetric actor-critic

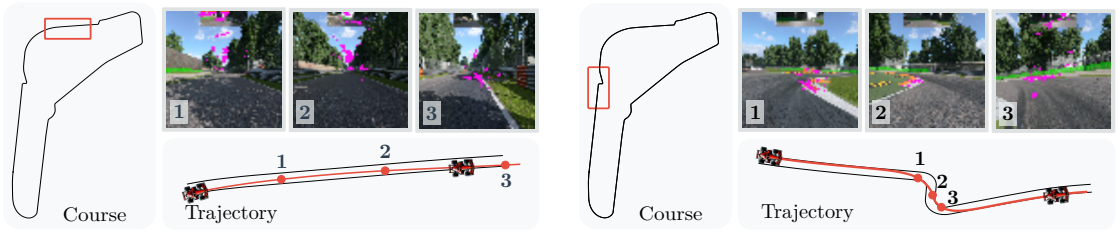


Figure 6: GGC visualization of our racing agent for two sections of the `Monza` track: (left) a straight section; (right) a chicane section. (Top) We show in pink the positive gradients for the delta steering angle action computed using the policy of our agent. We show the top 80% of the gradients in the visualization, to reduce noise. We highlight two different behaviors: in straight sections, our agent focuses on far-away visual features, such as tree lines (Left: 1, 2) and distinct far-away shades (Left: 3); in chicane sections, our agent focuses on close elements that are fundamental to effectively change direction, changing its focus from the apex of the immediate curve (Right: 1, 2) to the curb on the opposite side (Right: 3). Best viewed with color and zoomed in.

architecture that uses global features from the simulator to train accurate critic functions, while the policy function only uses local features to output actions at execution time. We demonstrated that our approach surpassed the fastest human lap time in three time trial scenarios and showed comparable performance to super-human methods that require global features for its policy. We hope our proposed approach helps to build the foundations for a novel research field on competitive autonomous racing agents with car-centric input features.

For future work, we consider three research threads to address the limitations of this paper. First, we plan on extending our approach to racing scenarios with multiple vehicles in the track, in order to allow vision-based autonomous agents to race against human drivers in the same track. Second, although we showed that our asymmetric architecture allows us to train super-human agents with a simple deep RL training setup, we still use propriocentric information as inputs of our agent. To relax this necessity, we will explore incorporating recurrent neural networks, similarly to Wadekar et al. (2021). Finally, we plan to add generalization capabilities to our agent, which deals with conditions unseen during training. We can extend our training setup to include various tracks and car models with additional image data augmentations (Kostrikov et al., 2020) to mitigate this issue, and eventually transfer the trained agent to real-world racing vehicles.

Broader Impact Statement

We focused on evaluating our agent in a high-fidelity simulator in this paper. However, our research can also contribute to the development of real-world end-to-end autonomous race cars. Using car-centric inputs, agents can control vehicles without using external localization systems that usually require domain knowledge beforehand or expensive engineering costs to design. By extending our agent to real world setups, we could simplify the pipeline of autonomous vehicles.

Acknowledgments

We are very grateful to Polyphony Digital Inc. and Sony Interactive Entertainment for enabling this research. Furthermore, we would like to thank our colleagues in Sony AI, Songyang Han, Hojoon Lee, Craig Sherstan, Florian Fuchs, and Patrick MacAlpine, for their feedback on this manuscript. The first author also acknowledges that this work has been partially supported by the European Research Council (ERC-BIRD), Swedish Research Council and Knut and Alice Wallenberg Foundation.

References

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Ex-

plainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

Andrea Baisero, Brett Daley, and Christopher Amato. Asymmetric dqn for partially observable reinforcement learning. In *Uncertainty in Artificial Intelligence*, pp. 107–117. PMLR, 2022.

Johannes Betz, Hongrui Zheng, Alexander Liniger, Ugo Rosolia, Phillip Karle, Madhur Behl, Venkat Krovi, and Rahul Mangharam. Autonomous vehicles on the edge: A survey on autonomous vehicle racing. *IEEE Open Journal of Intelligent Transportation Systems*, 3:458–488, 2022.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Peide Cai, Hengli Wang, Huaiyang Huang, Yuxuan Liu, and Ming Liu. Vision-based autonomous car racing using deep imitative reinforcement learning. *IEEE Robotics and Automation Letters*, 6 (4):7262–7269, 2021.

Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pp. 1–16. PMLR, 2017.

Florian Fuchs, Yunlong Song, Elia Kaufmann, Davide Scaramuzza, and Peter Dürr. Super-human performance in gran turismo sport using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(3):4257–4264, 2021.

Manan S Gandhi, Bogdan Vlahov, Jason Gibson, Grady Williams, and Evangelos A Theodorou. Robust model predictive path integral control: Analysis and performance guarantees. *IEEE Robotics and Automation Letters*, 6(2):1423–1430, 2021.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

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.

Ce Hao, Chen Tang, Eric Bergkvist, Catherine Weaver, Liting Sun, Wei Zhan, and Masayoshi Tomizuka. Outracing human racers with model-based autonomous racing. *arXiv preprint arXiv:2211.09378*, 2022.

James Herman, Jonathan Francis, Siddha Ganju, Bingqing Chen, Anirudh Koul, Abhinav Gupta, Alexey Skabelkin, Ivan Zhukov, Max Kumskoy, and Eric Nyberg. Learn-to-race: A multimodal control environment for autonomous racing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9793–9802, 2021.

Thomas Herrmann, Francesco Passigato, Johannes Betz, and Markus Lienkamp. Minimum race-time planning-strategy for an autonomous electric racecar. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6. IEEE, 2020.

Ryuji Imamura, Takuma Seno, Kenta Kawamoto, and Michael Spranger. Expert human-level driving in gran turismo sport using deep reinforcement learning with image-based representation. *arXiv preprint arXiv:2111.06449*, 2021.

Maximilian Jaritz, Raoul De Charette, Marin Toromanoff, Etienne Perot, and Fawzi Nashashibi. End-to-end race driving with deep reinforcement learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 2070–2075. IEEE, 2018.

Pierre-Alexandre Kamienny, Kai Arulkumaran, Feryal Behbahani, Wendelin Boehmer, and Shimon Whiteson. Privileged information dropout in reinforcement learning. *arXiv preprint arXiv:2005.09220*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

John Lambert, Ozan Sener, and Silvio Savarese. Deep learning under privileged information using heteroscedastic dropout. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8886–8895, 2018.

Nan Li, Eric Goubault, Laurent Pautet, and Sylvie Putot. Autonomous racecar control in head-to-head competition using mixed-integer quadratic programming. In *Opportunities and challenges with autonomous racing, 2021 ICRA workshop*, 2021.

Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1):18, 2020.

Alexander Liniger and John Lygeros. A viability approach for fast recursive feasible finite horizon path planning of autonomous rc cars. In *Proceedings of the 18th international conference on hybrid systems: computation and control*, pp. 1–10, 2015.

Alexander Liniger and John Lygeros. A noncooperative game approach to autonomous racing. *IEEE Transactions on Control Systems Technology*, 28(3):884–897, 2019.

Xueguang Lyu, Andrea Baisero, Yuchen Xiao, Brett Daley, and Christopher Amato. On centralized critics in multi-agent reinforcement learning. *Journal of Artificial Intelligence Research*, 77:295–354, 2023.

Yaqub Mahmoud, Yuichi Okuyama, Tomohide Fukuchi, Tanaka Kosuke, and Iori Ando. Optimizing deep-neural-network-driven autonomous race car using image scaling. In *SHS web of conferences*, volume 77, pp. 04002. EDP Sciences, 2020.

Federico Massa, Luca Bonamini, Alessandro Settimi, Lucia Pallottino, and Danilo Caporale. Lidar-based gnss denied localization for autonomous racing cars. *Sensors*, 20(14):3992, 2020.

David Q Mayne, María M Seron, and SV Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

Gennaro Notomista, Mingyu Wang, Mac Schwager, and Magnus Egerstedt. Enhancing game-theoretic autonomous car racing using control barrier functions. In *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 5393–5399. IEEE, 2020.

Matthew O'Kelly, Varundev Sukhil, Houssam Abbas, Jack Harkins, Chris Kao, Yash Vardhan Pant, Rahul Mangharam, Dipshil Agarwal, Madhur Behl, Paolo Burgio, et al. F1/10: An open-source autonomous cyber-physical platform. *arXiv preprint arXiv:1901.08567*, 2019.

Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

Matthew O'Kelly, Hongrui Zheng, Achin Jain, Joseph Auckley, Kim Luong, and Rahul Mangharam. Tunercar: A superoptimization toolchain for autonomous racing. In *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 5356–5362. IEEE, 2020.

Wen-zheng Peng, Yin-hui Ao, Jing-hui He, and Peng-fei Wang. Vehicle odometry with camera-lidar-imu information fusion and factor-graph optimization. *Journal of Intelligent & Robotic Systems*, 101:1–13, 2021.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*, 2017.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

Adrian Remonda, Sarah Krebs, Eduardo Veas, Granit Luzhnica, and Roman Kern. Formula rl: Deep reinforcement learning for autonomous racing using telemetry data. *arXiv preprint arXiv:2104.11106*, 2021.

Ines Rito Lima, Shlomi Haar, Lucas Di Grassi, and A Aldo Faisal. Neurobehavioural signatures in race car driving: a case study. *Scientific reports*, 10(1):11537, 2020.

Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, et al. Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International conference on intelligent transportation systems (ITSC)*, pp. 1–6. IEEE, 2020.

Sasha Salter, Dushyant Rao, Markus Wulfmeier, Raia Hadsell, and Ingmar Posner. Attention-privileged reinforcement learning. In *Conference on Robot Learning*, pp. 394–408. PMLR, 2021.

Wilko Schwarting, Alyssa Pierson, Sertac Karaman, and Daniela Rus. Stochastic dynamic games in belief space. *IEEE Transactions on Robotics*, 37(6):2157–2172, 2021.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Aman Sinha, Matthew O'Kelly, Hongrui Zheng, Rahul Mangharam, John Duchi, and Russ Tedrake. Formulazero: Distributionally robust online adaptation via offline population synthesis. In *International Conference on Machine Learning*, pp. 8992–9004. PMLR, 2020.

Amit Sinha and Aditya Mahajan. Asymmetric actor-critic with approximate information state. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 7810–7816. IEEE, 2023.

Kieran Strobel, Sibo Zhu, Raphael Chang, and Skanda Koppula. Accurate, low-latency visual perception for autonomous racing: Challenges, mechanisms, and practical solutions. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1969–1975. IEEE, 2020.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. 2018.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Peter M Van Leeuwen, Stefan De Groot, Riender Happee, and Joost CF De Winter. Differences between racing and non-racing drivers: A simulator study using eye-tracking. *PLoS one*, 12(11): e0186871, 2017.

José L Vázquez, Marius Brühlmeier, Alexander Liniger, Alisa Rupenyan, and John Lygeros. Optimization-based hierarchical motion planning for autonomous racing. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2397–2403. IEEE, 2020.

Shakti N Wadekar, Benjamin J Schwartz, Shyam S Kannan, Manuel Mar, Rohan Kumar Manna, Vishnu Chellapandi, Daniel J Gonzalez, and Aly El Gamal. Towards end-to-end deep learning for autonomous racing: On data collection and a unified architecture for steering and throttle prediction. *arXiv preprint arXiv:2105.01799*, 2021.

Grady Williams, Brian Goldfain, Paul Drews, Kamil Saigol, James M Rehg, and Evangelos A Theodorou. Robust sampling based model predictive control with sparse objective information. In *Robotics: Science and Systems*, volume 14, pp. 2018, 2018.

Peter R Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, et al. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 602(7896): 223–228, 2022.

Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.

A Extended Related Work

Autonomous Racing (AR) is a subfield of autonomous driving research (Yurtsever et al., 2020) that concerns autonomous vehicles that operate at their dynamical and power limits within racing environments (such as racing circuits) (Betz et al., 2022).

Classical Approaches for AR Research in autonomous vehicle racing can traditionally be categorized into three different sub-areas: perception, planning and control. In *perception*, the overarching goal is to enable high-frequency object detection, mapping and localization while the vehicle is driving around the track at high-speeds: Massa et al. (2020) propose a LIDAR-based localization system that exploits a previously built map of the track environment to provide localization, achieving an accuracy of two meters when the car is moving at 200 km/h; Peng et al. (2021) contributes a multimodal odometry method (image, LIDAR, IMU) using factor-graph optimization to localize the vehicle in the track; Strobel et al. (2020) use YOLOV3 (Redmon & Farhadi, 2018) to detect light cones in the limits of the racing track for Formula Student competitions. In *planning*, the overarching goal is to plan spatial and velocity trajectories (global and local) that minimize lap time across the track: Herrmann et al. (2020) formulate an optimization-based velocity planner as a multi-parametric sequential quadratic problem that can handle a spatial and time variable friction coefficient; Vázquez et al. (2020) propose a hierarchical controller for autonomous racing, where the high-level controller computes the optimal trajectory in the race track (raceline) and the low-level controller attempts to follow the precomputed optimal trajectory; other approaches attempt to plan high-level behavior (such as overtaking maneuvers, or energy management during a race) either by assigning plans to a specific cost function and selecting the plan with the lowest overall cost (Liniger & Lygeros, 2015; Sinha et al., 2020; O'Kelly et al., 2020) or by combining the planner with game theoretical methods (Notomista et al., 2020; Schwarting et al., 2021; Liniger & Lygeros, 2019). In *control*, the overarching goal is to develop methods that are able to maintain the vehicle as close as possible to the planned spatial trajectory and speed profile. For this purpose, model predictive control (MPC) methods are widely employed: Williams et al. (2018) propose a robust sampling-based MPC framework based on a combination of model predictive path integral control and nonlinear Tube-MPC (Mayne et al., 2005), highlighting the framework's robustness in a real-world autonomous racing task; Gandhi et al. (2021) contribute a novel architecture for robust model predictive path integral control (RMPPI) and investigate its performance guarantees, highlighting its applicability in a real-world off-road navigation task; Li et al. (2021) propose a nonlinear MPC model under a minimum time objective, which integrates the opponent vehicle's trajectory as a collision-avoidance constraint, to allow racing tasks with opponents. In our paper, contrary to all previous works, we explore end-to-end RL for autonomous racing vehicles that combines the pipeline of perception-planning-control into a single process.

Deep Neural Networks for AR Recent developments in deep neural networks (DNN) have allowed the development of end-to-end methods that are able to learn to race directly from observations. Wadekar et al. (2021) explore different types of data collection techniques to train DNNs to output steering and throttle actions in a supervised learning manner. Mahmoud et al. (2020) highlight that reducing the image size in CNN-based networks leads to an increase in the performance of a DNN-based racing method both in simulation and in the real-world. Contrary to these works, we focus in RL approaches that learn to perform racing tasks through trial-and-error interaction with the environment.

Reinforcement Learning for AR RL has also been shown to be a promising approach to learn suitable racing behavior, motivated in part by the development of realistic driving simulators that are able to model the dynamics of the car and of the track (O'Kelly et al., 2019; Herman et al., 2021; Rong et al., 2020; Dosovitskiy et al., 2017). Methods that employ RL to train racing agents often provide both image observations and additional features, that can be either propriocentric (e.g., velocity, acceleration) or related to the track (e.g., center line of the track). Jaritz et al. (2018) explore vision-based RL for driving agents in the context of a rally game. The propose an Asynchronous Advantage Actor-Critic (A3C) (Mnih et al., 2016) architecture that exploits both visual

information (first-person camera images) and propriocentric features (velocity and acceleration) to learn to race in the game environment. The authors show that their approach is able to generalize to unseen tracks and highlight the importance of initializing the agents at random checkpoints during training for the performance of the method. However, the authors state that their method does not achieve optimal racing trajectories, "lacking anticipation", and is unable to complete the racing tracks without colliding several times with obstacles. Recently, Cai et al. (2021) described an approach that merges imitation learning and model-based reinforcement learning to learn autonomous racing agents. Central to their contribution is the Reveries-Net architecture to learn a probabilistic world model (Ha & Schmidhuber, 2018). The authors propose an iterative training procedure after pretraining the policy with expert demonstrations: (i) learn the world model with the current dataset of experiences, (ii) refine the policy using rollouts from the world model and (iii) collect new data using the refined policy in the environment. The authors evaluate their approach in simulation and real-world racing environments and demonstrate that their method outperforms previous imitation learning and RL methods in sample efficiency and performance. However, their method requires expert-level demonstrations to pretrain the policy and cannot be trained only from interaction with the environment. Despite their reported ability to consistently drive around the track, the racing performance of these methods is still sub-optimal: some works lack a performance comparison against humans (Remonda et al., 2021; Jaritz et al., 2018) or, when such comparison is made, the methods still under perform significantly against median human users (Cai et al., 2021; Herman et al., 2021): for example, Cai et al. (2021) reports a 10 second gap to the lap time of a normal human user. Imamura et al. (2021) also explores vision-based RL for racing agents. The authors propose to pretrain an image encoder on observations of the track environment collected by a random policy and, subsequently, use the frozen encoder during policy training. However, their method can only achieve expert-level performance in time trial tasks, still reporting a three second difference to the best human players. In this work, we contribute a novel vision-based RL agent that consistently outperforms the best human drivers in a racing task.

Super-human Racing Performance in AR Recently, super-human performance of autonomous racing RL agents have been reported by Fuchs et al. (2021) and Wurman et al. (2022). Fuchs et al. (2021) introduced a model-free RL approach and designed a novel proxy reward that considers the progress of the agent in the course. The method is able to achieve super-human performance in time trial tracks in Gran Turismo Sport (GTS), a highly realistic racing simulation. Moreover, the authors show that their approach generates trajectories that are qualitatively similar to the ones recorded by the best human drivers, highlighting high-level racing behavior (such as in-out driving along curves). More recently, Wurman et al. (2022) introduced Gran Turismo Sophy (GT Sophy) agent, a RL agent that is able to achieve super-human performance both in time trial and racing tasks with multiple opponents in GTS. To achieve this level of performance, the authors contribute a novel asynchronous distributional actor-critic algorithm (QR-SAC) using multiple training scenarios (e.g., with different number of opponents, with randomized positions and speeds). Furthermore the authors designed a novel reward function that accounts for track-related behavior (e.g., progress in the course, off-course racing) and for event-related behavior (e.g., overtaking opponents or being overtaken, colliding with opponents). The authors show that GT Sophy is able to exhibit tactical skills that allow it to beat expert humans in head-to-head racing. However, to achieve super-human performance both approaches require *global features*, such as forward looking course shape information. In this work, we contribute the first RL agent that is able to achieve super-human performance in a racing task using only car-centric local features during execution.

Asymmetric Reinforcement Learning Recent works have explored asymmetrical training architectures for reinforcement learning to mitigate partial observability during execution time (Pinto et al., 2017; Salter et al., 2021; Kamienny et al., 2020; Sinha & Mahajan, 2023; Baisero et al., 2022). Pinto et al. (2017) explored asymmetrical training in the context of learning policies in simulation for robotic systems that are transferable to real-world setups. To do so, the authors design an asymmetrical actor-critic training scheme in which the critic is provided with the state of the simulation environment and the policy is provided with RGBD information. Furthermore, the authors introduce domain randomization during training in the simulator, showing that it improves the robustness of

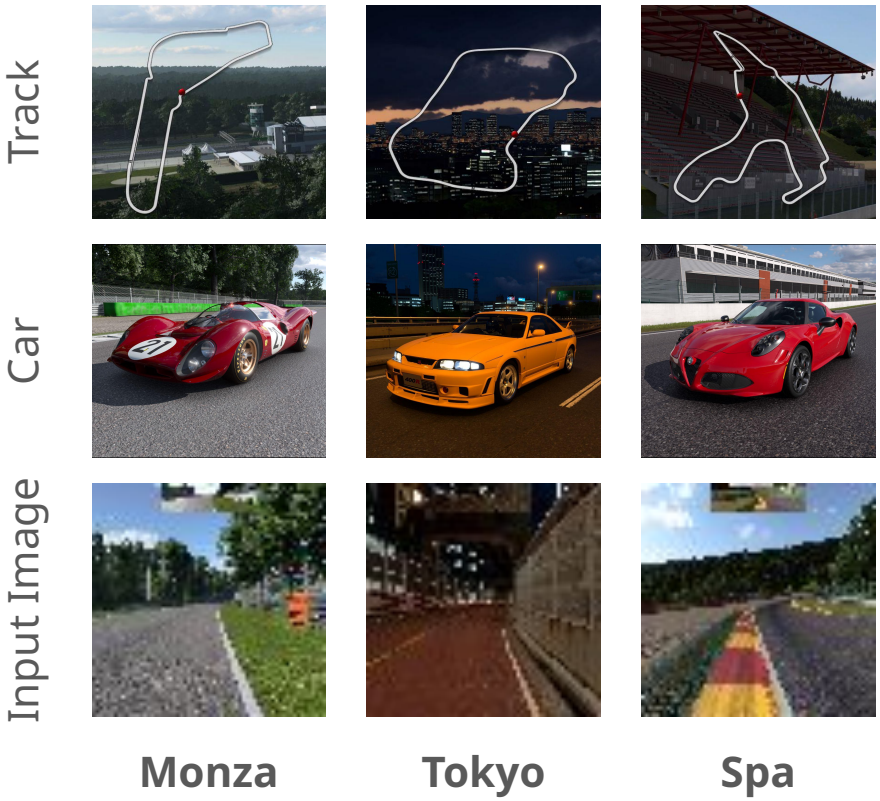


Figure 7: Description of the tracks, cars and examples of image observations used to train and evaluate our racing agent. We rescale all images to 64×64, without any modification. We turn off all HUD information. The top notch in images is a rear mirror view of the vehicle.

the learned policy to distractor elements during execution in real-world experiments. However, domain randomization can often lead to an increase the complexity of the learning process, impacting the overall performance of the agent. To address this issue Salter et al. (2021) propose to train two actor-critic agents that share experiences: one that is provided with state information and another that is provided with image information. Furthermore, the authors introduce an attention mechanism in each agent that is aligned throughout training. The authors show that the attention-based asymmetrical training scheme improves the efficiency and the robustness of the learning process. In contrast to prior work, Kamienny et al. (2020) explores providing privileged information (PI) to both the critic and the policy networks using dropout. In particular, the authors evaluate the use of PI-Dropout (Lambert et al., 2018) in the context of RL and show how it outperforms other methods to exploit privileged information, such as distillation or auxiliary losses, in scenarios with partial observability. In the previous works the asymmetrical training scheme is posited experimentally, without theoretical guarantees on the convergence of the algorithms. Recent works have studied the theoretical properties of asymmetrical reinforcement learning: Baisero et al. (2022) introduced an asymmetrical version of policy iteration and of the Q-learning algorithm with convergence guarantees; Sinha & Mahajan (2023) proposed an asymmetrical version of the actor-critic algorithm and derive performance bounds on their algorithm. Asymmetric training has also been explored in the context of multi-agent reinforcement learning (MARL), in part to deal with the decentralized nature of executing policies for multiple agents (Oliehoek et al., 2008; Rashid et al., 2020; Sunehag et al., 2017; Lyu et al., 2023). Approaches in cooperative MARL often exploit the paradigm of Centralized Training with Decentralized Execution (CTDE): during training, agents have access to a centralized critic that exploits the joint observation of all agents; at execution time each agent can only exploit

its own observation to run their policy (Oliehoek et al., 2008; Sunehag et al., 2017; Rashid et al., 2020).

In this work, we also explore an asymmetrical actor-critic training scheme, where we provide global features to the critic, allowing the execution of the policy only with local features. We show how our approach enables learning super-human racing agents.

B Additional Details of the Evaluation Scenarios

In this section we present additional details regarding our evaluation scenarios. In Figure 7 we present the track layouts, the cars employed and examples of image observations across the three scenarios. We highlight that our agent achieves super-human performance across a wide variety of car dynamics, track layouts and conditions.

C Additional Ablation Studies

C.1 Course Point Range

We evaluate how the range of the course point feature affects the performance of our agent. We reduce the range to 2 seconds and 4 seconds, denoted as *low* and *medium course points* respectively. Figure 8 shows the result of comparing our agent to the variations with different range of course points. We observe that the shorter range of course points makes performance unreliable. On the other hand, using medium course point range slightly improves the performance of our agent. In this paper, we used the same range of course points as the one described in Wurman et al. (2022). However, this result indicates that tuning this hyperparameter could provide an additional performance improvement to our agent. We leave the tuning of this parameter for future work.

C.2 Synchronous Training and Execution

As discussed in Appendix I, to train our agents we employ a synchronous training and testing scheme, similar to other simulators such as OpenAI Gym (Brockman et al., 2016), where the simulator only executes simulation steps after receiving the next action commands sent by a rollout worker. However, by default, GT7 executes its simulation asynchronously in real time. We evaluate the feasibility of learning and executing policies in an asynchronous setting. Figure 8 shows the result of comparing our agent, which trains and executes only on an synchronous mode, to the variations which: (i) train in synchronous mode and executes in asynchronous mode (*Sync/Async*) and (ii) train and executes in asynchronous mode (*Async/Async*). The results highlight the importance of the synchronous training, as training our agent with asynchronous mode results in a significant decrease in performance: this variation is only able to outperform the best human drivers only in 7.45% of the laps. However, the performance of the policy of the agents that train in synchronous mode and execute asynchronously does not decrease significantly and is still able to outperform the best human drivers in 69.3% of the laps. For future work, we plan on exploring parameter-efficient neural networks and optimizing hardware setups to mitigate the effect of latency on the control of our agents.

C.3 Training with Masked Rear-view Mirror

In Section 5.2 we show that our agent considers information in the rear-view mirror section of the image observation to race across the task. To understand whether the rear-view mirror is fundamental to achieve the super-human performance, we additionally train our agents without rear-view mirror information, by masking this section with black pixels. Figure 8 shows that our agent can still consistently achieve the super-human lap time even without the rear-view mirror. Based on this experimental result, we conclude that the attention to the rear-view mirror is an artifact of our end-to-end training scheme on a track with a static layout.

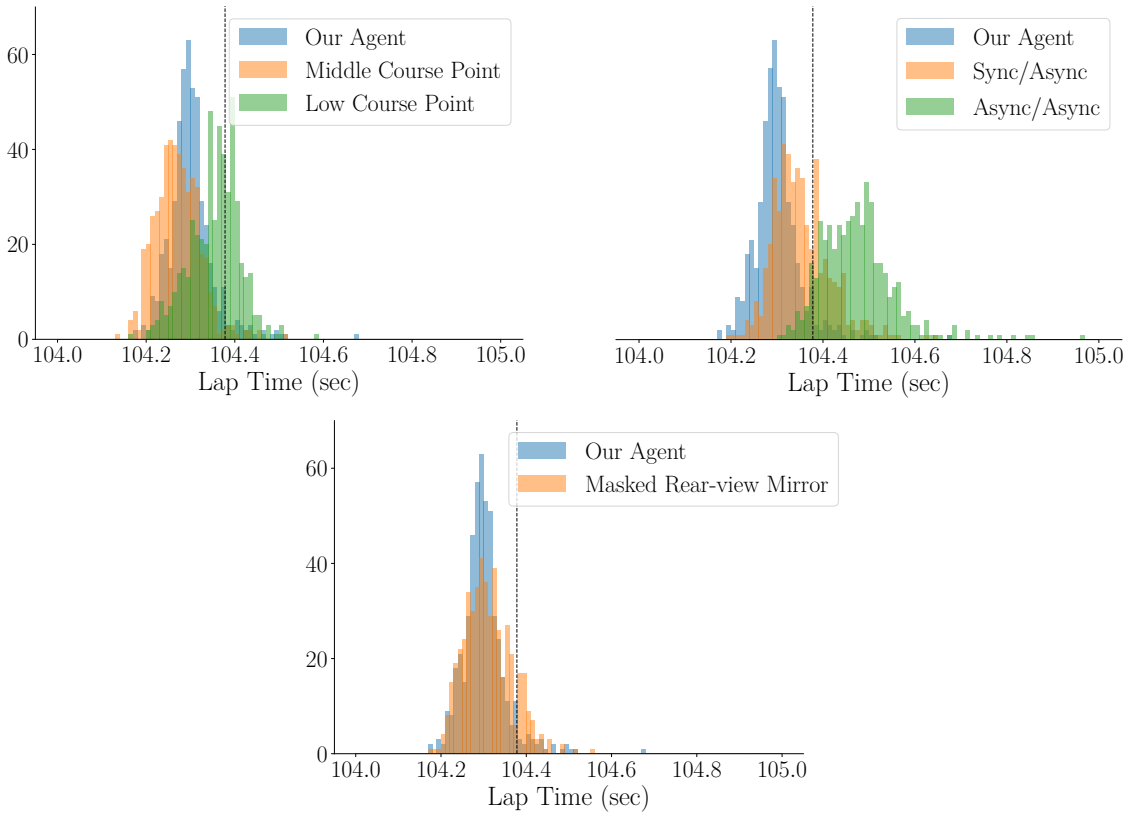


Figure 8: Performance study of our racing agent in the `Monza` scenario in regards to (top left) different range of course points, (top right) different training/test synchronicity conditions and (bottom) different setups in the rear-view mirror. Additionally, we highlight the lap time of the fastest human player (black dashed line). We consider five randomly-seeded training runs and show the distribution of 500 evaluation laps, with 100 laps executed by the fastest policy in each training run. Lower is better.

D Training Curves

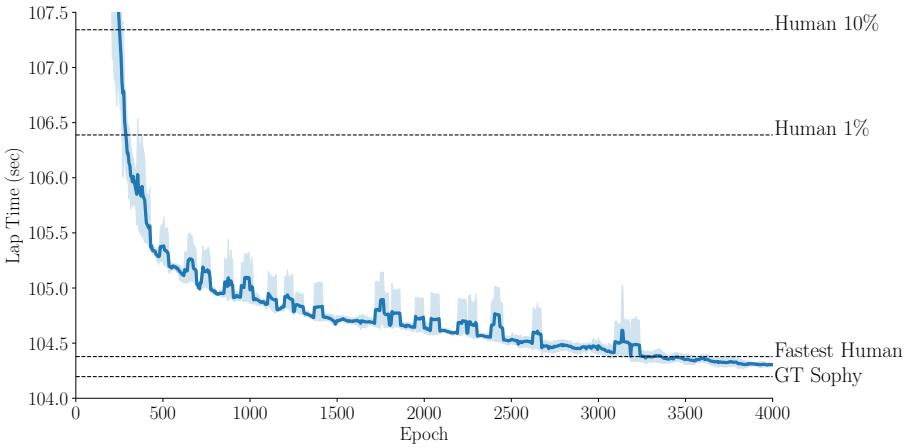
We present in Figure 9 the overall training curves of our agent across all scenarios. The results show that our agent quickly exceeds top 1% human performance and is able to consistently surpass the fastest human lap time. At the end of training, our agent also achieved comparable performance to GT Sophy. We also present the overall training curves of our agent across all ablation conditions of Section 5.1 and Appendix C in Figure 10.

E Additional Trajectory Analysis

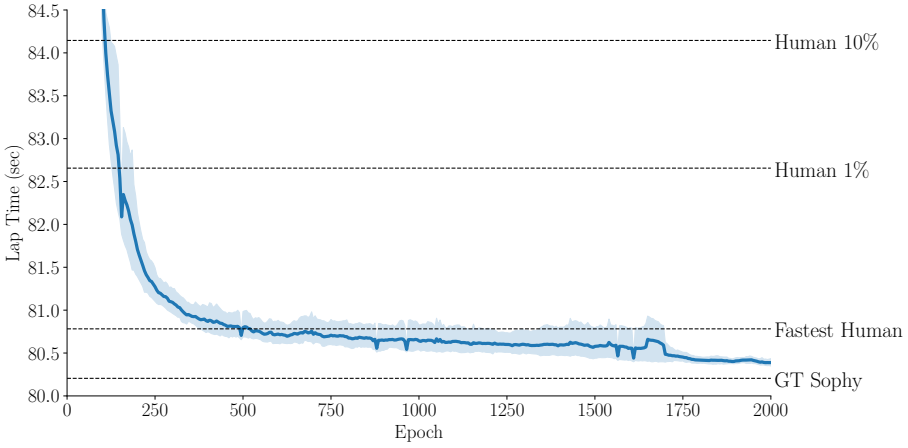
E.1 Comparison to Fastest Human Driver

We qualitatively compare the trajectories of our agent and of the best human player across all scenarios: `Monza` in Figure 11, `Tokyo` in Figure 12 and `Spa` in Figure 13.

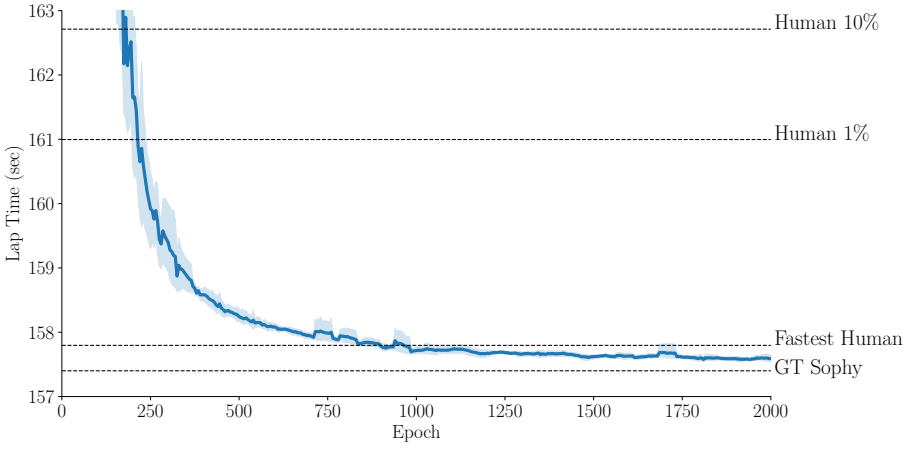
The results show, across multiple sections of the tracks, that our agent does not simply follow the trajectory of the fastest human player but, in fact, exhibits novel racing behavior: in `Monza`, in straights our agent drives much closer to the curb, while the human player takes a more center line along the track, and in curves and chicanes it takes different driving lines; in `Tokyo` we once again see that our agent takes driving lines much closer to the curb than the fastest human; in `Spa`



(a) Monza

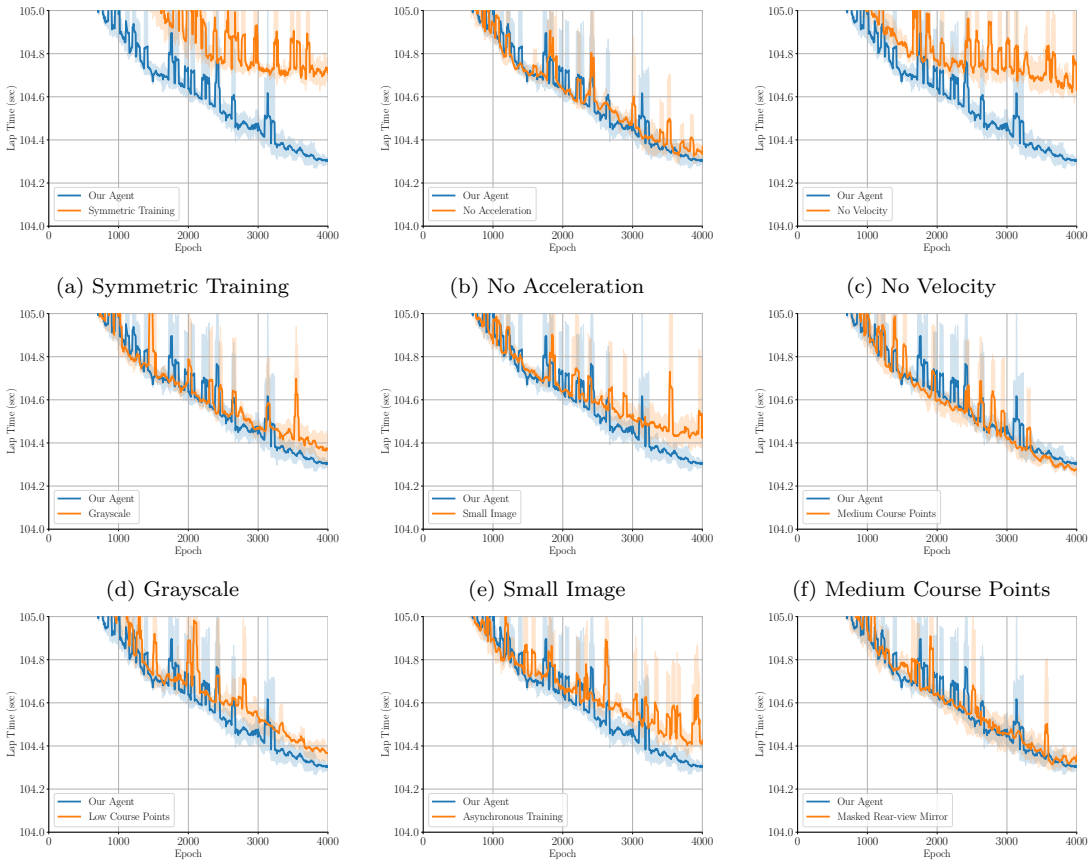


(b) Tokyo



(c) Spa

Figure 9: Training curves of our agent across all scenarios. We present the lap time per training epoch averaged over five randomly-seeded runs, with 95% confidence interval. We compare our agent against human performance (10%, 1% and fastest) and GT Sophy. Training curves are smoothed for visual clarity.



(a) Symmetric Training (b) No Acceleration (c) No Velocity

(d) Grayscale (e) Small Image (f) Medium Course Points

(g) Low Course Points (h) Asynchronous Training (i) Masked Rear-view Mirror

Figure 10: Training curves of our agent across all ablation conditions of Section 5.1 and Appendix C. We present the lap time per training epoch averaged over five randomly-seeded runs, with 95% confidence interval. Training curves are smoothed for visual clarity.

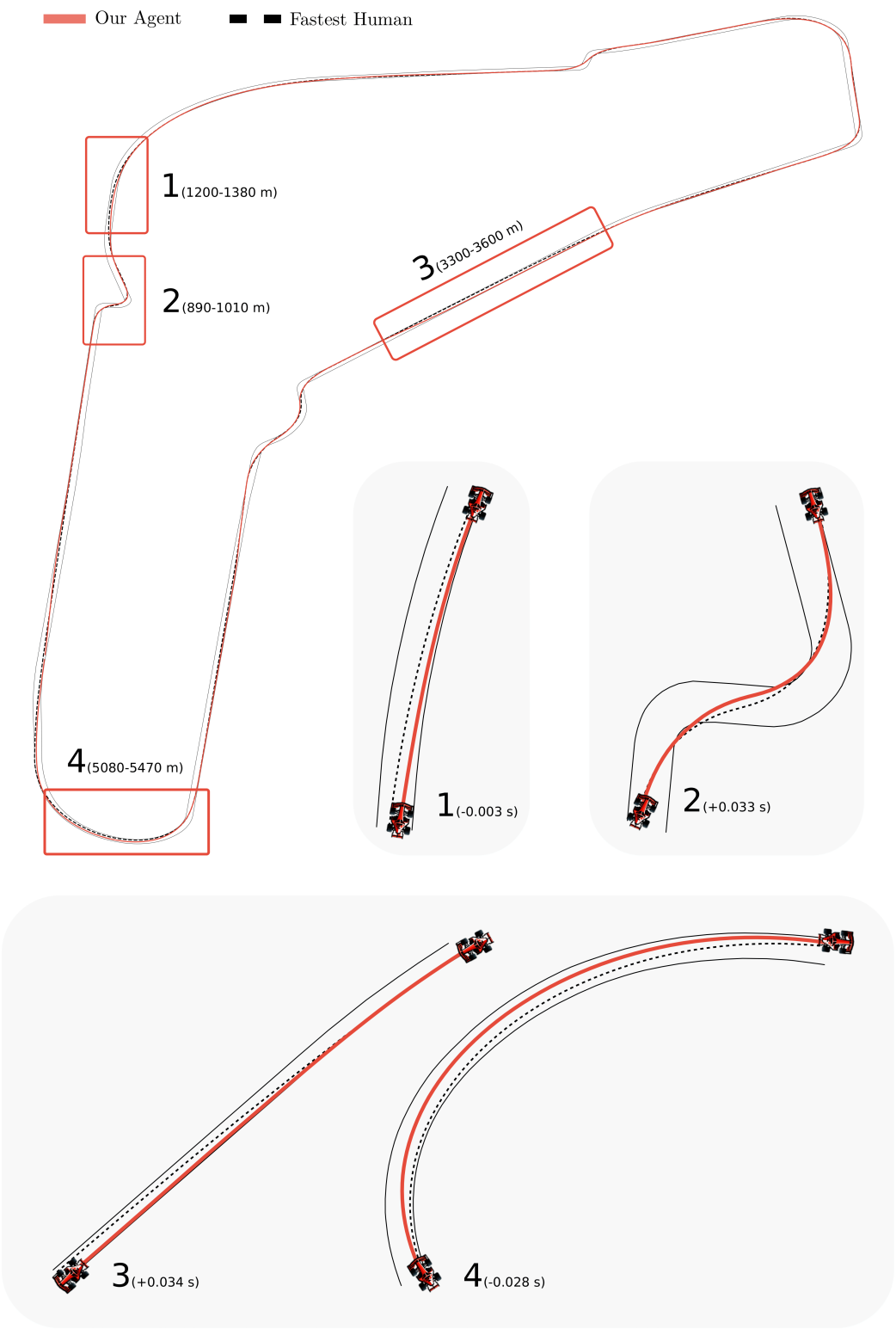


Figure 11: Trajectory comparison between our agent and the fastest human player in the `Monza` track. We highlight this comparison on (1, 3) straight sections approaching a curve, (2) a chicane section and (4) a curve section. We show in the figure the course progression of each segment as well as the time gained (negative values) or lost (positive values) to the best human driver. Best viewed zoomed in.

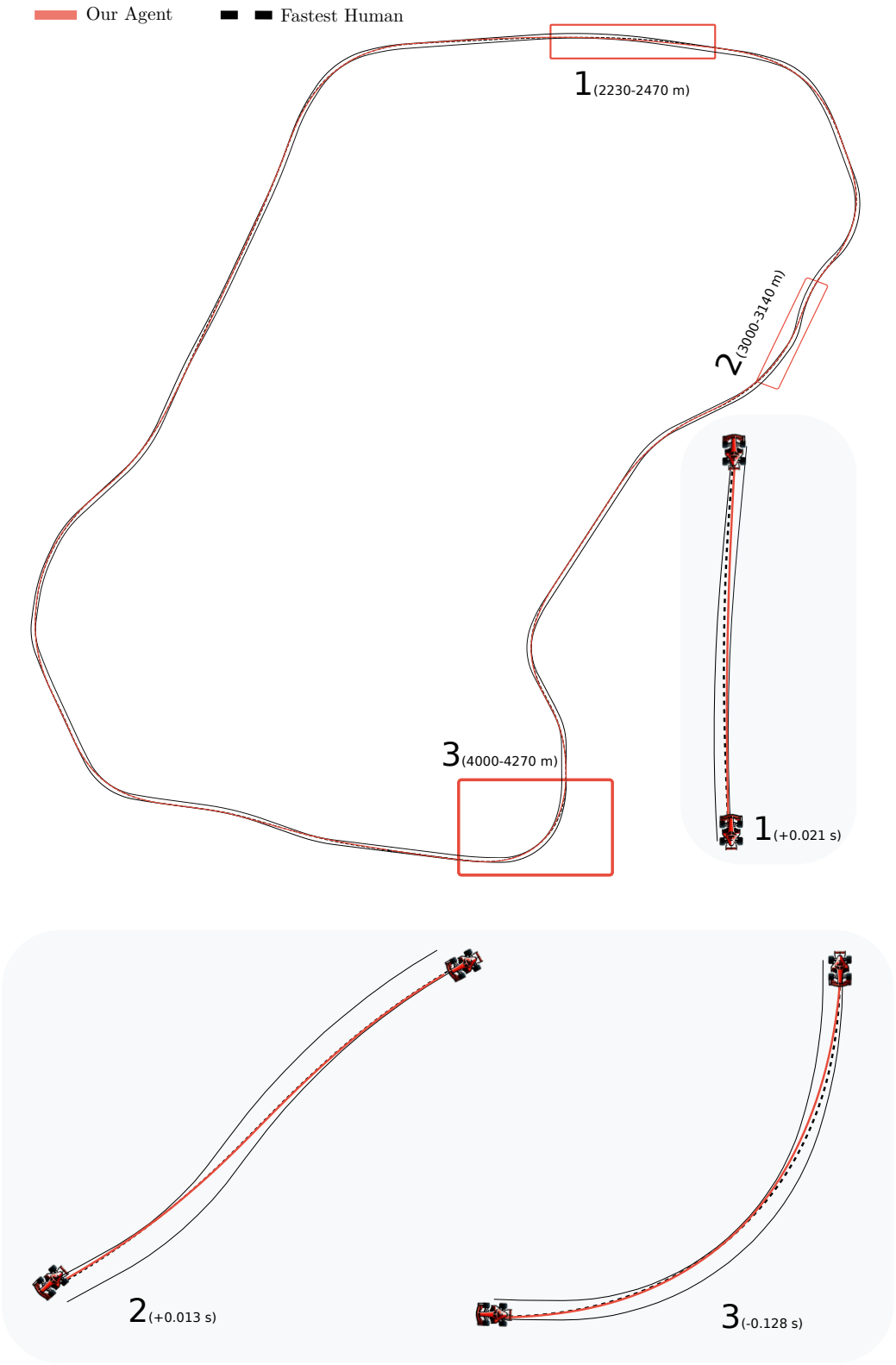


Figure 12: Trajectory comparison between our agent and the fastest human player in the `Tokyo` track. We highlight this comparison on (1, 2) straight sections and (3) a curve section. We show in the figure the course progression of each segment as well as the time gained (negative values) or lost (positive values) to the best human driver. Best viewed zoomed in.

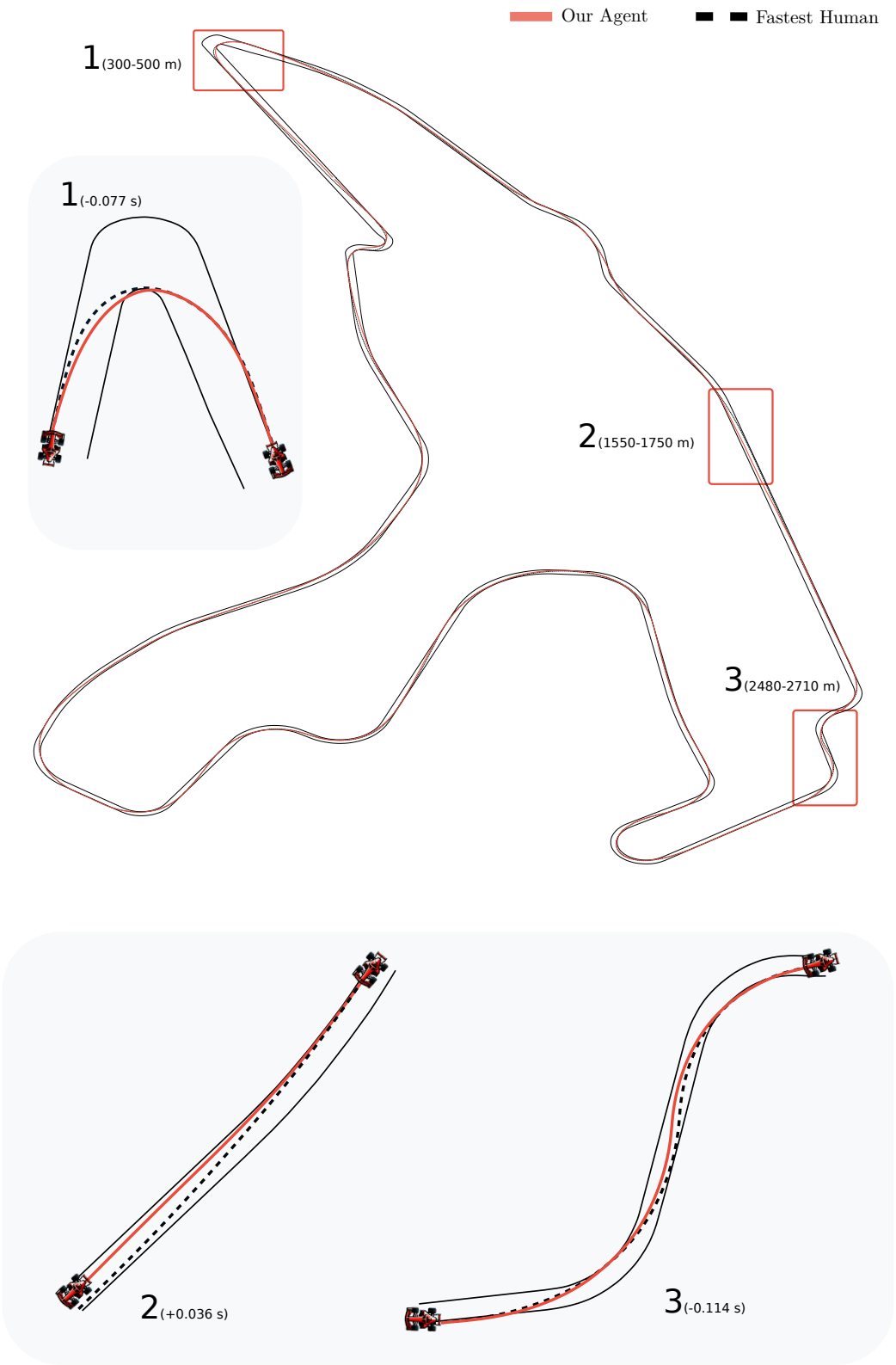
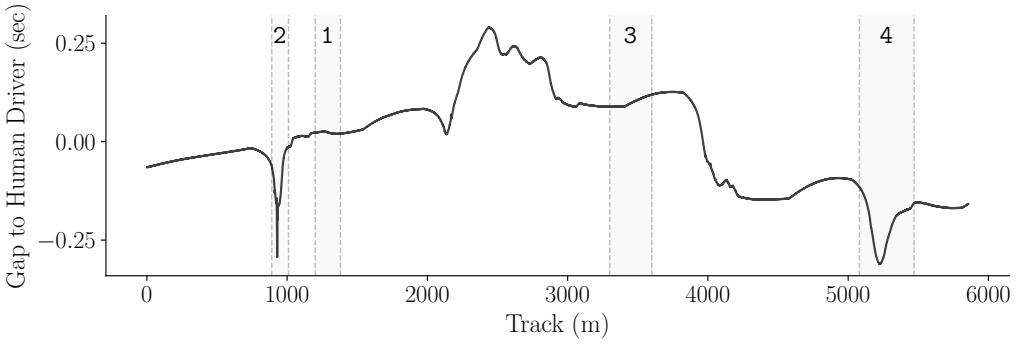
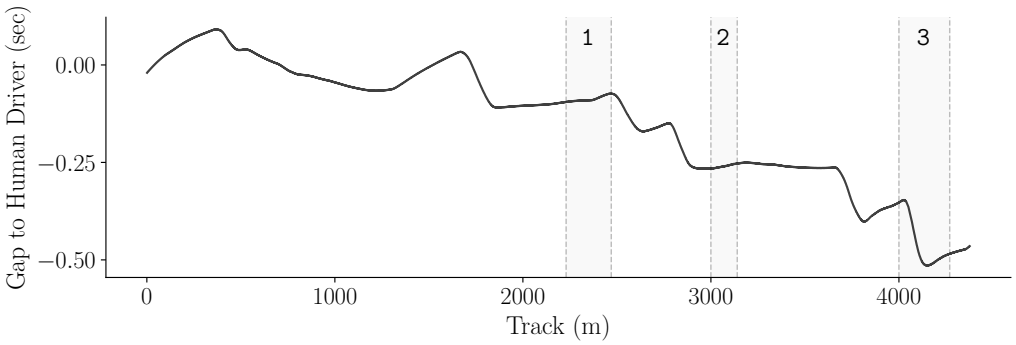


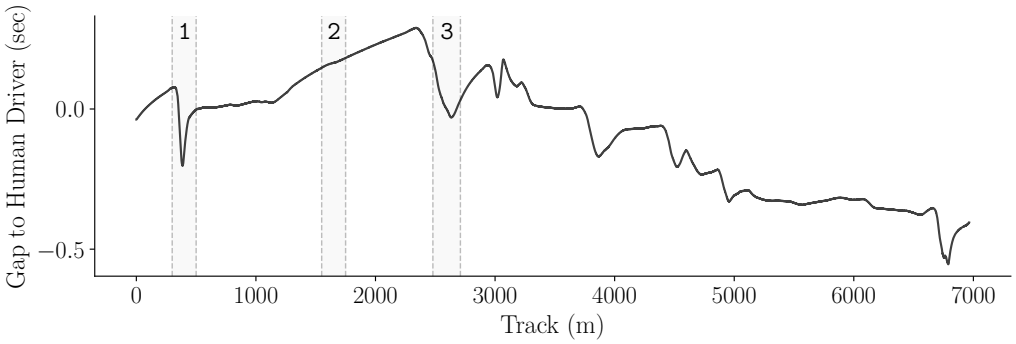
Figure 13: Trajectory comparison between our agent and the fastest human player in the `Spa` track. We highlight this comparison on (1) a U-turn section, (2) a straight section and (3) a chicane section. We show in the figure the course progression of each segment as well as the time gained (negative values) or lost (positive values) to the best human driver. Best viewed zoomed in.



(a) `Monza`



(b) `Tokyo`



(c) `Spa`

Figure 14: Time difference between our agent and the best human reference driver as a function of the progression in the track. We identify the trajectory sections highlighted in the `Monza` (Figure 11), `Tokyo` (Figure 12) and `Spa` (Figure 13) tracks. Lower is better.

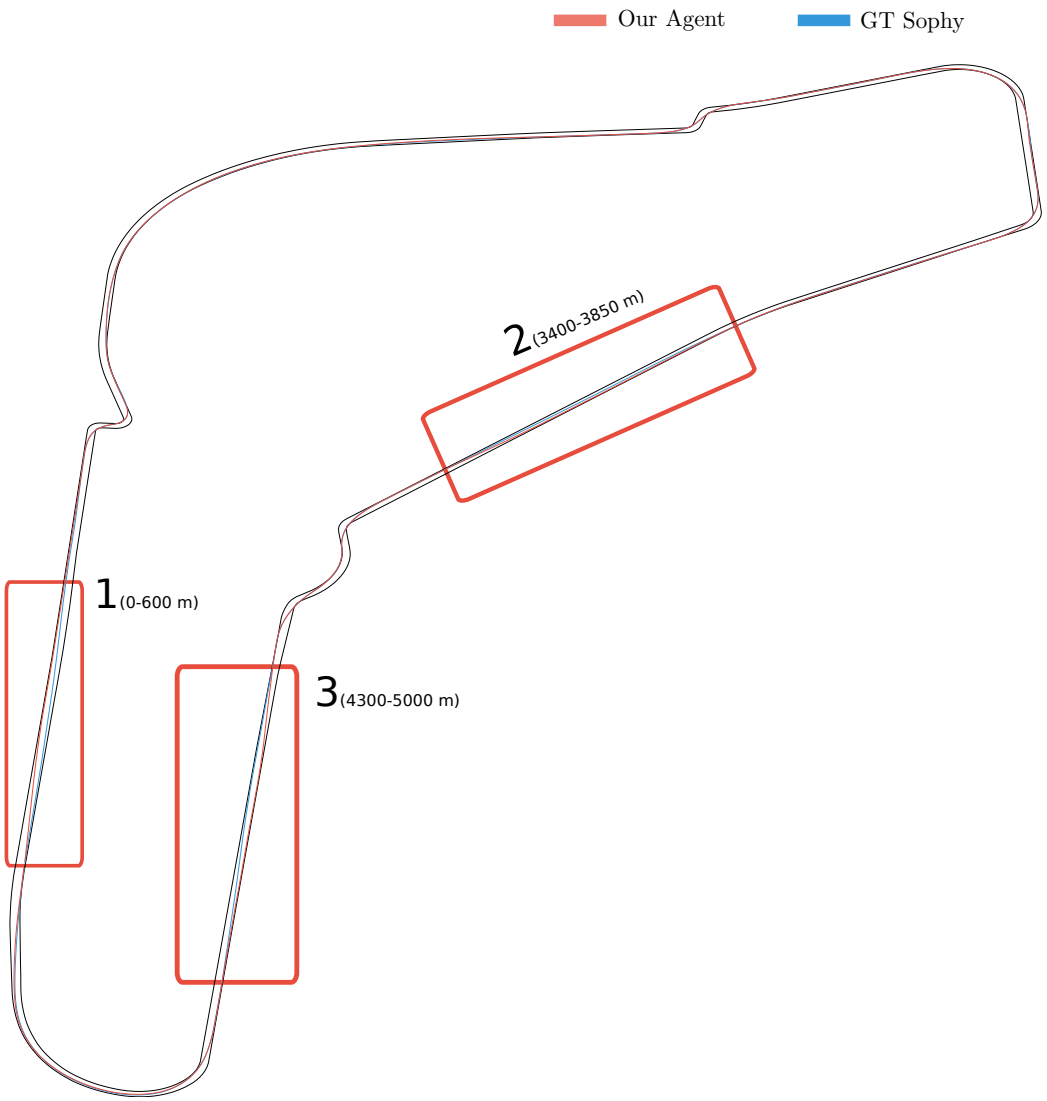


Figure 15: Trajectory comparison between our agent and GT Sophy (Wurman et al., 2022) in the `Monza` track. We highlight this comparison on sections that our trajectory is significantly different from Sophy's (1, 3) and sections where our agent loses time to Sophy (2). Best viewed in color and zoomed in.

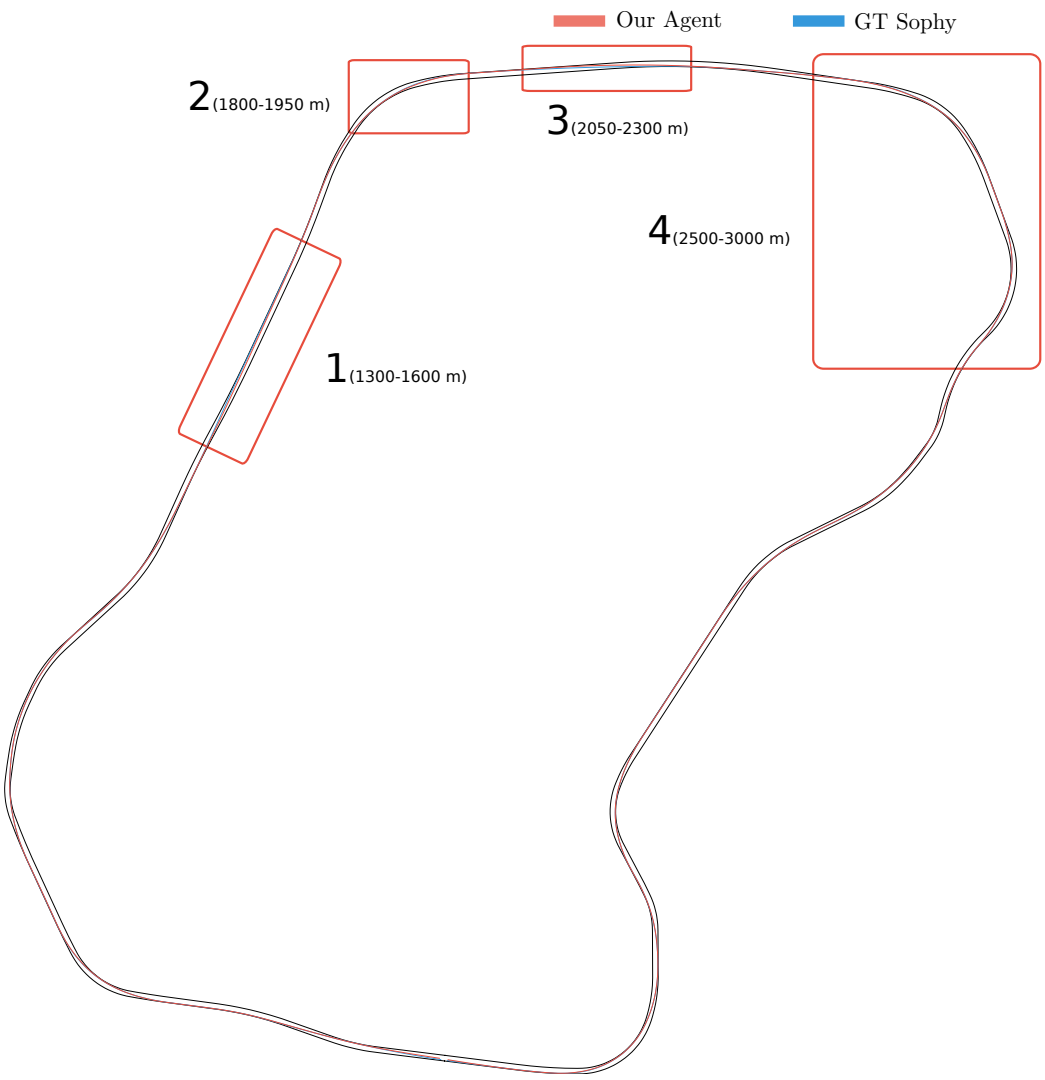


Figure 16: Trajectory comparison between our agent and GT Sophy (Wurman et al., 2022) in the `Tokyo` track. We highlight this comparison on sections where our trajectory is significantly different from GT Sophy's (1, 3) and sections where our agent loses time to Sophy (2, 4). Best viewed in color and zoomed in.

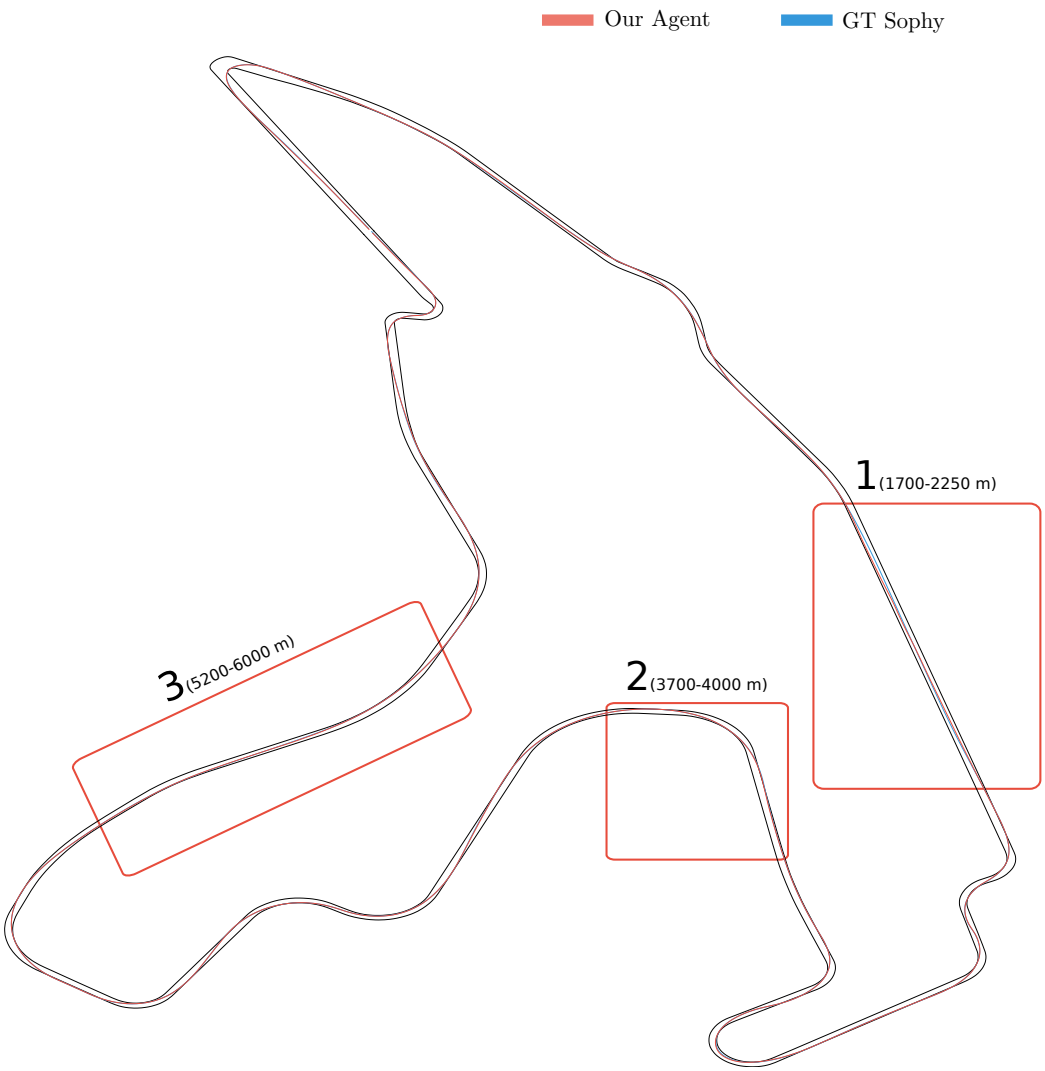
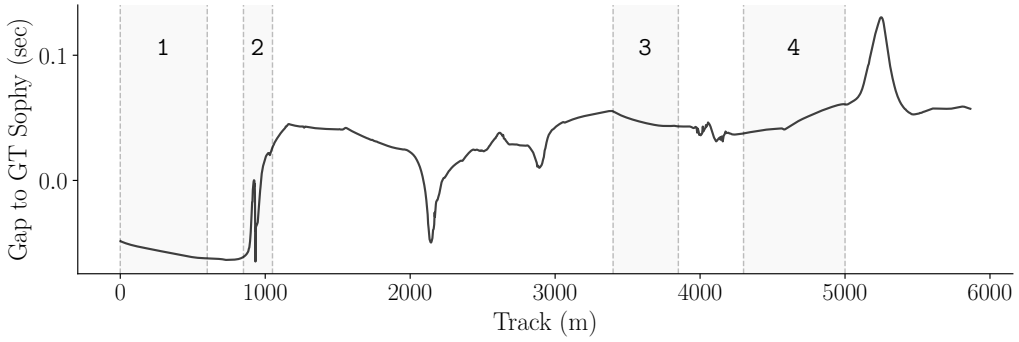
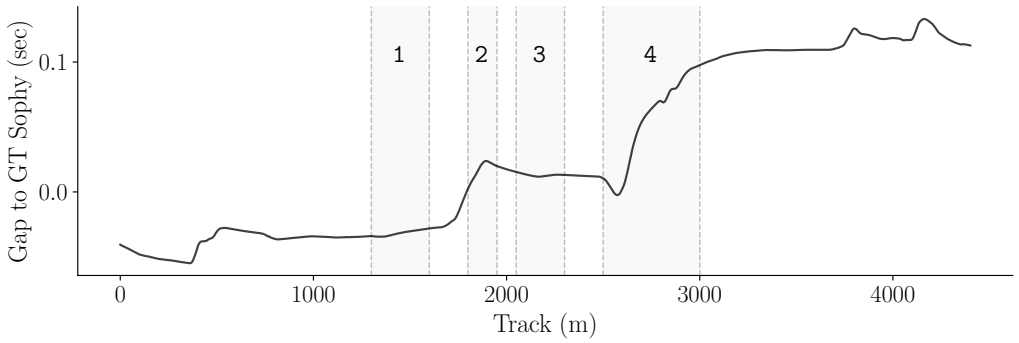


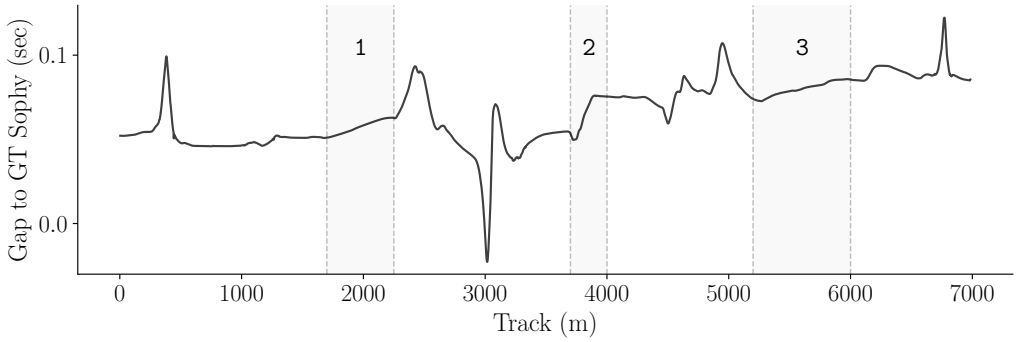
Figure 17: Trajectory comparison between our agent and GT Sophy (Wurman et al., 2022) in the `Spa` track. We highlight this comparison on sections where our trajectory is significantly different from GT Sophy's (1) and sections where our agent loses time to Sophy (2, 3). Best viewed in color and zoomed in.



(a) `Monza`



(b) `Tokyo`



(c) `Spa`

Figure 18: Time difference between our agent and GT Sophy as a function of the progression in the track. We identify the trajectory sections highlighted in the `Monza` (Figure 15), `Tokyo` (Figure 16) and `Spa` (Figure 17) tracks. Lower is better.

the results show that our agent takes different driving lines in both straights and curves. These significant differences in driving behavior raise the potential to use super-human racing agents as a training tool for human drivers, as previously identified in (Wurman et al., 2022).

We also present the time difference between our agent and the fastest human reference driver along the progression of the track in Figure 14. The results show the competitive nature and challenge of our task: our agent does not simply outperforms the human driver from the beginning of the lap, but gains and loses time against the human driver throughout the whole track. However, by the end of the track, our agent is able to outperform the fastest human driver.

E.2 Comparison to GT Sophy

We qualitatively compare the trajectories of our agent and of GT Sophy (Wurman et al., 2022) across all scenarios: `Monza` in Figure 15, `Tokyo` in Figure 16 and `Spa` in Figure 17. The results show that overall the trajectory of our agent follows that of Sophy, despite our agent not having access to global features. However, the results show that often in long straights, for example Figure 15 (1, 3), our agent takes a different racing line to Sophy, due to the absence of long-range information about the track in our agent's input.

Additionally, in Figure 18 we present the time difference between our agent and GT Sophy along the progression of the track, to understand where our agent actively loses time. The results show that, despite the similarity of the trajectories in these sections, our agent mostly loses time in curve sections, for example Figure 17 (2, 4). This result hints that GT Sophy, having access to long-range, precise information about the forward track limits, can approach the curve with a better velocity profile, leading to the sudden increase in the time difference between our agent and GT Sophy.

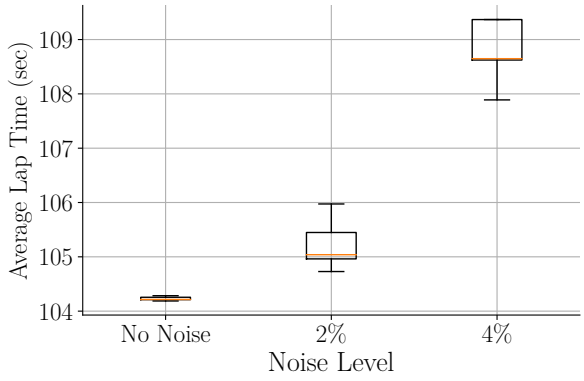
F Perturbation Study

To identify the limitations of our agent, we conducted an additional evaluation under various perturbation conditions. In this section, we select the top five agents from each `Monza` trial, for all evaluation. Note that training an agent able to generalize to unseen conditions during training is outside the scope of this work. For completeness, we present here a robustness evaluation and leave the further improvement of the generalization of our agent to future work.

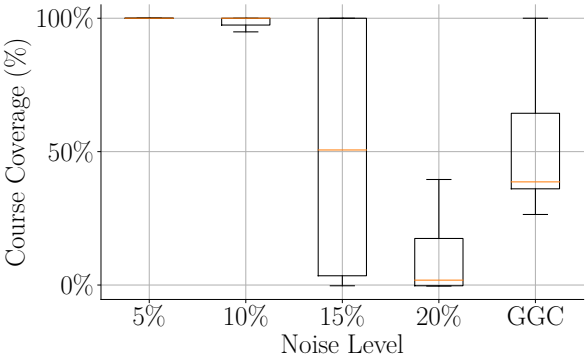
Noise in propriocentric observations: We evaluated our agent under different levels of noise in the propriocentric observations by following the same procedure of Fuchs et al. (2021). All agents completed laps with 2% noise and 33% of the agents completed laps with 4% noise. However, the agents are no longer able to drive with more than 6% noise. Figure 19a shows the average lap times of the completed laps, which suggests that the agent loses racing performance with increased noise levels in the propriocentric observation.

Noise in image observations: We evaluated our agent under different levels of noise in the image observations. The noise level is defined as the percentage of randomly selected pixels from the complete image that are replaced by black pixels at each time step. We additionally evaluated the same type of noise applied to the pixels highlighted by GGC analysis, as described in Section 5.2, which we denote as GGC. Note that GGC, on average, highlights up to 5% of the total number of pixels of the image in our setting. Figure 19b shows the course progression achieved by the agents with each noise level. The agents consistently completed laps with up to 10% noise and didn't complete any laps at 20% noise. Interestingly, the agents dropped performance significantly with GGC-based noise even though GGC only highlights 5% of pixels at most. This result further indicates that the pixels highlighted by GGC are vital for our agent to control the vehicle.

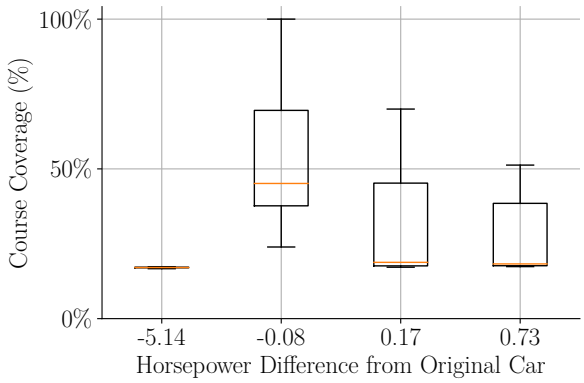
Generalization to different time-of-the day conditions: GT7 allows us to change the time of the race, resulting in a change in environmental elements, such as position and number of clouds, but also on the lighting conditions of the track. We then use the time of the race as a feature to evaluate the robustness of our agent against image perturbations. We observe that evaluating our



(a) Noise in propriocentric observations



(b) Noise in image observations



(c) Generalization to different cars

Figure 19: Results of the perturbation study of our agent in the `Monza` scenario: (a) Completed lap times of agents with noise in the propriocentric features added during evaluation; (b) Course coverage of agents with noise in the image features added during evaluation; (c) Course coverage of agents with different car models during evaluation.

agents at different times than the one of training results in their inability to complete laps around the track.

Generalization to different cars: To test the robustness of our agent to unseen dynamics, we evaluate it with different car models from the one used during training. We sorted all car models available in GT7 by horsepower and selected four *neighbor* car models (two slower cars and two faster cars)[2]. Figure 19c shows the course progression achieved by each car model. The result suggests that executing our policy in cars with increasingly higher horsepower leads to the worse generalization capability: due to their higher performance (e.g. in terms of velocity), the agent may experience dynamical states not experienced during training (e.g., high velocities in straight sections). On the other hand, the results show that our agent is comparably better in driving lower-performance cars.

G Guided Grad-CAM for Visual Analysis of Policies

We adapt the basic Guided Gradient-weighted Class Activation Mapping (GGC) (Selvaraju et al., 2017) algorithm to RL scenarios, following:

1. During execution, we perform a forward pass given image and propriocentric observations $(\mathbf{o}^i, \mathbf{o}^p)$. The policy network outputs an action-specific average value a, from a truncated Gaussian distribution $\mathcal{N}(a, a_\sigma)$.
2. We compute the gradient of the mean a with respect to the feature maps A^k in the last convolutional layer of the image encoder (Conv4 in Table 2):
$$\frac{\partial a}{\partial A^k}.$$
3. We apply a global average pooling to the Grad-CAM gradient to obtain neuron importance weights w_a^k for each feature map:
$$w_a^k = \frac{1}{Z} \sum_i \sum_j \frac{\partial a}{\partial A_{ij}^k},$$
where $Z = W \times H$ is the normalization factor for the spatial dimensions of the feature map, W is the weight of the feature map and H is the height of the feature map.
4. We obtain the Grad-CAM activation map, $L^{\text{Grad-CAM}}$ by performing a weighted combination of the feature maps followed by a ReLU function:
$$L_a^{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k w_a^k A^k\right).$$
5. We perform a separate backpropagation step to compute the gradient of the action value a with respect to the input image observation $\mathbf{o}^i$:
$$\frac{\partial a}{\partial \mathbf{o}^i}.$$
Furthermore, we filter out negative values from this gradient, in order to consider only features that have a positive influence on the action.
6. We multiply the Grad-CAM activation map with the guided backpropagation result to obtain the Guided Grad-CAM visualization:
$$L_a^{\text{Guided Grad-CAM}} = L_a^{\text{Grad-CAM}} \times \frac{\partial a}{\partial \mathbf{o}^i}.$$
7. Finally, we normalize the GGC visualization and, possibly, clip the result by a predefined noise threshold value.

[2]The selected car models are Porsche 959 '87, Peugeot 205 Turbo 16 Evolution 2 '86, Dodge Viper GTS '02, and Lamborghini Countach 25th Anniversary '88, in ascending order of horsepower.

H Additional Guided Grad-CAM visualizations

We present additional GGC visualizations for the action *delta steering angle* for our agent across all tracks: `Monza` (Figure 20), `Tokyo` (Figure 21) and `Spa` (Figure 22). Once again, the results shows that our agent focuses on different features depending on the track section: in long straights, our agent focuses on far-away visual features, such as horizon of the track or tree lines in the distance, while in chicanes and tight curves, our agent focuses on the curbs of the track, which are fundamental to successfully change its direction without going off-track.

I Implementation Details

Model Architecture: We provide the detailed architecture of our agent in Table 2.

Data collection: At the beginning of every episode, the initial position of the agent is uniformly sampled from in-course areas as well as left and right off-course areas within 5% of track width. The agent faces towards a center line 30m ahead and the launch speed is uniformly sampled from 0 to 104.607km/h. We reset the episode every 150 seconds.

Distributed training: Unlike widely used RL simulators that can execute faster with powerful computing resources (e.g., MuJoCo (Todorov et al., 2012)), GT7 executes its simulation in real time. To compensate for the simulation speed, we use an asynchronous distributed training scheme by following Wurman et al. (2022). In this work, we consider 20 rollout workers for data collection and policy evaluation, each assigned to a different PlayStation® 4 system connecting to rollout workers via ethernet. However, the game screen retrieval via ethernet induces additional latency, which makes it difficult to train agents in real time. Therefore, we configured the simulator to block simulation steps until the simulator receives the next action commands sent by the rollout worker. We applied this setting to all experiments including the baseline GT Sophy. We evaluate the impact of the synchronous communication scheme on the performance of our agent in Appendix C.2.

J Reward and Training Hyperparameters

We present in Table 3 the reward function parameters used in our work, selected empirically. Moreover, we present in Table 4 the list of training hyperparameters used by our approach. We keep the same hyperparameters across all scenarios, except for the number of training epochs: in `Monza` we use 4000 training epochs and in both `Tokyo` and `Spa` we use 2000 epochs, where an epoch consists of 6000 gradient steps. We use a higher number of training epochs in `Monza` because in this scenario we employ a faster racing car, which requires a more precise maneuver to achieve a higher level of performance.

K Additional Details on Image Resolution

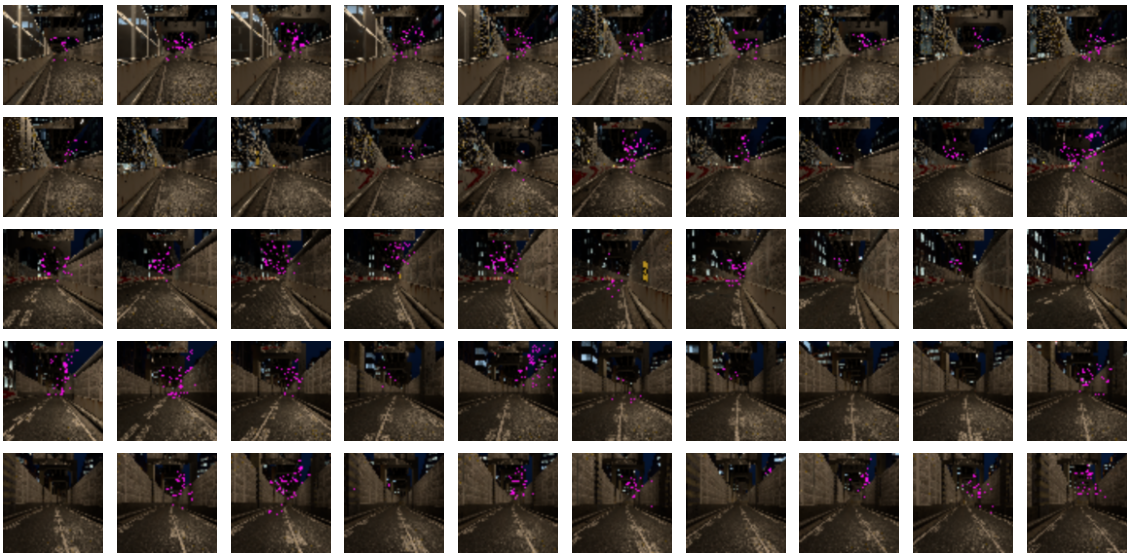
To highlight how image compression affects the elements in the observation of our agent, we provide examples of game images with different resolutions in Figure 23: the original 1920×1080 image, a 64×64 image used by our agent and a reduced 32×32 image used in our ablation study in Section 5.1. The figure shows that across all resolutions we can identify critical elements of the environment to perform the task (such as the track limits), yet with decreasing level of precision.

The choice of resolution also affects the total parameter count of our model: since we keep the same architecture for both our agent and the *small image* ablated version, the size of feature maps for the latter version in each convolutional layers is half the size of values of the former, described in Table 2. It may be possible to improve the performance of the *small image* ablation by optimizing the convolutional encoder architecture for input images of 32×32, but we leave this exploration to future work.

(a) Straight section (150-539 m)

(b) Chicane section (899-993 m)

Figure 20: Guided Grad-CAM (GGC) visualization of our racing agent for two sections of the `Monza` track: (Top) a straight section; (Bottom) a chicane section. We show in pink the positive gradients for the delta steering angle action computed using the policy of our agent. We show the top 80% of the gradients in the visualization, to reduce noise. Best viewed with color and zoomed in.



(a) Straight outdoor section (625-898 m)

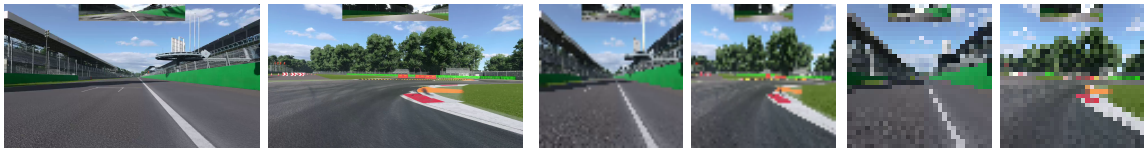
(b) Straight tunnel section (3499-3787 m)

Figure 21: Guided Grad-CAM (GGC) visualization of our racing agent for two sections of the `Tokyo` track: (Top) a straight outdoor section; (Bottom) a straight indoor section. We show in pink the positive gradients for the delta steering angle action computed using the policy of our agent. We show the top 80% of the gradients in the visualization, to reduce noise. Best viewed with color and zoomed in.

(a) Chicane section (2296-2484 m)

(b) Straight section (2741-2963 m)

Figure 22: Guided Grad-CAM (GGC) visualization of our racing agent for two sections of the `Spa` track: (Top) a chicane section; (Bottom) a straight section. We show in pink the positive gradients for the delta steering angle action computed using the policy of our agent. We show the top 80% of the gradients in the visualization, to reduce noise. Best viewed with color and zoomed in.



(a) 1920×1080 (Native screen resolution) (b) 64×64 (Our Agent) (c) 32×32 (*Small Image*)

Figure 23: Example observations with three different resolutions: in images with 64×64 resolution we can still observe some level of detail in most objects; However, in images with 32×32 we are able to recognize only basic elements of the environment (e.g., track limits, sky, background).

Table 2: Model architecture of our racing agent.

Layer Description	Input Dimensions	Output Dimensions
Actor Network		
Conv1: 64 filters, 4x4, stride 2, ReLU	64x64x3	32x32x64
Conv2: 128 filters, 4x4, stride 2, ReLU	32x32x64	16x16x128
Conv3: 256 filters, 4x4, stride 2, ReLU	16x16x128	8x8x256
Conv4: 512 filters, 4x4, stride 2, ReLU	8x8x256	4x4x512
FC, 128 units, ReLU	4x4x512	128
MLP FC1: 2048 units, ReLU	145 (128 + 17)	2048
MLP FC2: 2048 units, ReLU	2048	2048
MLP FC3: 2048 units, ReLU	2048	2048
MLP FC4: 2048 units, ReLU	2048	2048
MLP FC Output: 4 units, `Tanh`	2048	4
Critic Network		
MLP FC1: 2048 units, ReLU	531	2048
MLP FC2: 2048 units, ReLU	2048	2048
MLP FC3: 2048 units, ReLU	2048	2048
MLP FC4: 2048 units, ReLU	2048	2048
MLP FC Output: 32 units, linear	2048	32

Table 3: Reward parameters of our racing agent.

Parameter	Value
λ^o	10
λ^w	10
λ^s	3
λ^h	5
c^d	0.014
c^s	182.883569
c^o	0.034

Table 4: Training hyperparameters of our racing agent.

Hyperparameter	Value
Activation function	ReLU
Optimizer	Adam (Kingma & Ba, 2014)
Batch size	512
Policy learning rate	2.5×10^{-5}
Critic learning rate	2.5×10^{-5}
Global norm of critic gradient clipping	10
Discount factor	0.9896
SAC entropy temperature (Haarnoja et al., 2018)	0.01
Number of quantiles	32
Multi-step	7
Replay buffer size	2.5M