electronics

MDPI

Article

Stable and Efficient Reinforcement Learning Method for Avoidance Driving of Unmanned Vehicles

Sun-Ho Jang [1,2], Woo-Jin Ahn [2], Yu-Jin Kim [2], Hyung-Gil Hong [1], Dong-Sung Pae [3,*] and Myo-Taeg Lim [2,*]

[1] Korea Institute of Robotics Technology Convergence, Andong 36728, Republic of Korea; jang1229@kiro.re.kr (S.-H.J.); honghg@kiro.re.kr (H.-G.H.)
[2] School of Electrical Engineering, Korea University, Seoul 02841, Republic of Korea; wjahn@korea.ac.kr (W.-J.A.); sally1004k@korea.ac.kr (Y.-J.K.)
[3] Department of Software, Sangmyung University, Cheonan 31066, Republic of Korea
[*] Correspondence: paeds915@smu.ac.kr (D.-S.P.); mlim@korea.ac.kr (M.-T.L.)

Abstract: Reinforcement learning (RL) has demonstrated considerable potential in solving challenges across various domains, notably in autonomous driving. Nevertheless, implementing RL in autonomous driving comes with its own set of difficulties, such as the overestimation phenomenon, extensive learning time, and sparse reward problems. Although solutions like hindsight experience replay (HER) have been proposed to alleviate these issues, the direct utilization of RL in autonomous vehicles remains constrained due to the intricate fusion of information and the possibility of system failures during the learning process. In this paper, we present a novel RL-based autonomous driving system technology that combines obstacle-dependent Gaussian (ODG) RL, soft actor-critic (SAC), and meta-learning algorithms. Our approach addresses key issues in RL, including the overestimation phenomenon and sparse reward problems, by incorporating prior knowledge derived from the ODG algorithm. With these solutions in place, the ultimate aim of this work is to improve the performance of reinforcement learning and develop a swift, stable, and robust learning method for implementing autonomous driving systems that can effectively adapt to various environments and overcome the constraints of direct RL utilization in autonomous vehicles. We evaluated our proposed algorithm on official F1 circuits, using high-fidelity racing simulations with complex dynamics. The results demonstrate exceptional performance, with our method achieving up to 89% faster learning speed compared to existing algorithms in these environments.

Keywords: reinforcement learning; meta learning; deep reinforcement learning; autonomous driving; robot operating system

check for updates

Citation: Jang, S.-H.; Ahn, W.-J.; Kim, Y.-J.; Hong, H.-G.; Pae, D.-S.; Lim, M.-T. Stable and Efficient Reinforcement Learning Method for Avoidance Driving of Unmanned Vehicles. *Electronics* **2023**, *12*, 3773. https://doi.org/10.3390/electronics12183773

Academic Editors: Peter Odry and Vladimir László Tadić

Received: 7 August 2023
Revised: 23 August 2023
Accepted: 29 August 2023
Published: 6 September 2023

1. Introduction

Reinforcement learning (RL) has recently gained notable attention in various fields, including autonomous driving, due to its capability to address unanticipated challenges in real-world scenarios. Autonomous driving software defects can pose potential risks, thus developing safe and efficient methods when using AI technologies for autonomous driving systems is important [1]. Autonomous driving systems, by employing RL algorithms, are able to accrue experience and refine their decision-making procedures within dynamic environments [2–4]. This can be largely attributed to RL's inherent ability to adapt and learn from complex and fluctuating situations, demonstrating its aptitude for these applications. The basic concept of RL lies in the structure of Markov decision processes (MDP), a system where algorithms learn via a trial-and-error approach, striving to reach predetermined objectives by learning from mistakes and rewards. The aim of RL is to optimize future cumulative rewards and formulate the most efficient policy for distinct problems [5]. Recent integration of deep learning with RL has demonstrated promising outcomes across various domains. This involves the employment of advanced neural networks such as convolutional neural networks (CNNs), multi-layer perceptrons, restricted Boltzmann machines, and recurrent neural networks [6,7]. By fusing reinforcement learning with deep learning, the system's learning capabilities are significantly enhanced, allowing it to process complex data such as sensor feedback and environmental observations, thus facilitating more informed and effective driving decisions [8]. However, the application of RL to autonomous driving presents a unique array of challenges, particularly when it comes to deploying RL in real-world environments. The uncertainties inherent in these environments can make the effective execution of RL quite challenging. As a result, researchers often struggle to achieve optimal RL performance directly within the actual driving context, highlighting the various obstacles encountered when applying RL to autonomous driving [9]. Several challenges plague the application of RL to autonomous driving: overestimation phenomenon, learning time, and sparse reward problems [10,11].

Firstly, the overestimation phenomenon is prevalent in model-free RL methods, such as Q-learning [12] and its variants like the double deep Q network (DDQN) [13,14] and dueling DQN [15]. These methods are susceptible to overestimation and incorrect learning, primarily due to the combination of insufficiently flexible function approximation and the presence of noise, which lead to inaccuracies in action values. Secondly, the significant amount of learning time required is another hurdle. When RL is fused with neural networks, it generates policies directly from interactions with the environment, bypassing the need for a basic dynamics model. However, even simple tasks necessitate extensive trials and a massive number of data for learning. This makes high-performance RL both time-consuming and data-intensive [16]. Lastly, the issue of sparse reward arises during RL training. This presents challenges in scenarios where not all conditions receive immediate compensation. Although techniques like hindsight experience replay (HER) [17,18] have been proposed to mitigate this issue, the direct application of RL to autonomous vehicles is still limited due to the complex fusion of information and potential system failures during the learning process. This paper addresses the challenges of RL in autonomous driving and reduces the reliance on extensive real-world learning by introducing a set of innovative techniques to enhance the efficiency and effectiveness of RL: data preprocessing through obstacle-dependent Gaussian (ODG) [19,20] DQN, prior knowledge through Guide ODG DQN, and meta-learning-based guided ODG DDQN.

The data preprocessing method employs the ODG algorithm to combat the overestimation phenomenon. By preprocessing distance information through ODG DQN, it allows for more accurate action values, fostering stable and efficient learning [21]. The prior knowledge method draws on human learning mechanisms, incorporating knowledge derived from the ODG algorithm. This strategy mitigates the issue of sparse rewards and boosts the learning speed [22], facilitating more effective convergence. Lastly, the meta-learning-based guide rollout method uses ODG DQN to address complex driving decisions and sparse rewards in real-world situations. By enriching prior knowledge using a rollout approach, this method aims to create efficient and successful autonomous driving policies.

Our main contributions can be summarized as follows:

- Efficiency and speed of learning: The newly proposed RL algorithm utilizes ODG DQN on preprocessed information, enabling the agent to make optimal action choices, which significantly enhances the learning speed and efficiency.
- Improvement of learning stability: With the use of prior knowledge, the guide-ODG-DQN helps mitigate the issue of sparse rewards, thus increasing the learning stability and overall efficiency.
- Adaptability to various environments: The meta-learning-based ODG DDQN leverages model similarities and differences to increase learning efficiency. This allows for the reliable training of a universal model across diverse environments, with its performance demonstrated in environments like Gazebo and Real-Environment.

In this context, the purpose and objectives of this study are to propose a stable and efficient reinforcement learning method to effectively address the overestimation phenomenon, learning time, and sparse reward problems faced in the field of autonomous driving. By doing so, we aim to improve the performance of reinforcement learning, overcome the obstacles for implementing autonomous driving systems in real environments, and provide more stable and efficient vehicle control strategies.

The remainder of this paper is organized as follows: in the stable and efficient method section, we mainly introduce the proposed reinforcement learning algorithm. To verify the effectiveness of our work, the experimental evaluations and necessary analysis are presented in the experiment. Finally, we summarize our work in the Conclusions section.

2. Stable and Efficient Reinforcement Learning Method

LiDAR (light detection and ranging) information serves as an invaluable perspective for autonomous driving systems, functioning much like a driver's sense by identifying obstacles through environmental analysis. LiDAR-based RL methods have found extensive application in research focused on judgement and control within autonomous driving systems such as the partially observable Markov decision process (POMDP) [23]. However, learning methodologies based on Q-learning, such as DDQN, encounter persistent overestimation issues, posing obstacles to the enhancement of learning efficiency and convergence speed.

To mitigate these issues, we propose a method that preprocesses and transforms the LiDAR value into valuable information attuned to the operating environment, implementing it as the ODG technique [24]. This approach, as depicted by the ODG module (in yellow) of Figure 1, is designed to reduce learning convergence time and boost efficiency by preprocessing RL input data, thus remedying scenarios with inaccurate action values. Furthermore, we introduce the concept of prior knowledge to address the sparse rewards issue that impedes RL's learning stability [25]. By integrating prior knowledge information from sparse reward sections, as demonstrated in the guide-ODG-DQN framework shown in the guide module (in blue) of Figure 1, we can enhance learning stability.

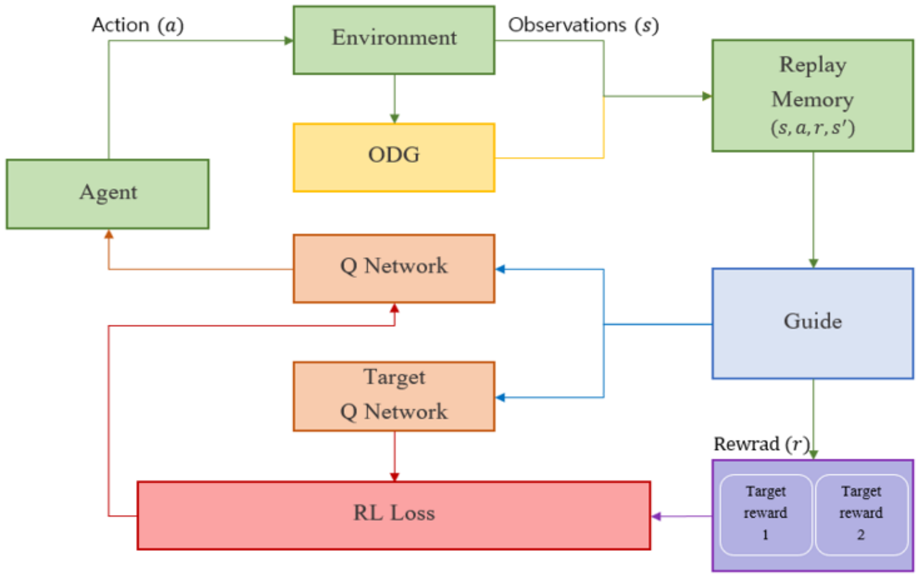


Figure 1. Process flow of stable and efficient reinforcement learning using proposed method.

It is noted that in RL, model performance can decline when the learning environment changes. Thus, we propose the meta-Guide ODG-DDQN method, represented in the target reward module (in purple) in Figure 1, to devise a more robust and adaptable RL algorithm. After training the model according to an initial goal, we modify the reward function to attain subsequent objectives. This approach effectively communicates the action value to the agent in diverse obstacle environments with reliability and swiftness. The proposed methodology consists of three progressively developed algorithms.

2.1. ODG DQN

Overestimation, a consequence of inaccurate action values, is underscored as a critical issue in the DDQN literature [13,26–28]. Traditional LiDAR information incorporates an infinite range, which represents all information at the maximum distance or the value of obstacle-free spaces. This arrangement leads to an overlap of LiDAR information within the system, causing overestimation and impeding the model's ability to select these infinite values. In Q-learning, this predicament can be defined by $Q(s,a) = V_*(s)$ for a given state s, as detailed in Equation (1). When environmental noise triggers an error, it is defined per Equation (2). If the max function is applied at the moment of peak value in Q-learning for action selection, the expression aligns with Equation (3). The bias, symbolized by $\sqrt{C/m-1}$, causes the model to overestimate the bias relative to the optimal value with Q-learning [12,13].

$$\sum_a (Q_t(s,a) - V_*(s)) = 0, \tag{1}$$

$$\frac{1}{m}\sum_a (Q_t(s,a) - V_*(s))^2 = C, \tag{2}$$

$$max_a Q_t(s,a) \geq V_*(s) + \sqrt{\frac{C}{m-1}}, \tag{3}$$

where m is the number of actions and C is a constant.

To address this overestimation, our algorithm utilizes the ODG module to preprocess state values. Illustrated in Figure 2, this module, based on Equation (4) with DQN [6], is engineered to establish an optimized steering angle model for the agent via Q-learning-based RL. This paves the way for the development of an optimized path plan built on the steering angle generated by the agent.

$$Q(s_t, a_t) = \mathbb{E}[r_t + \gamma max_a Q(s_{t+1}, a)]. \tag{4}$$

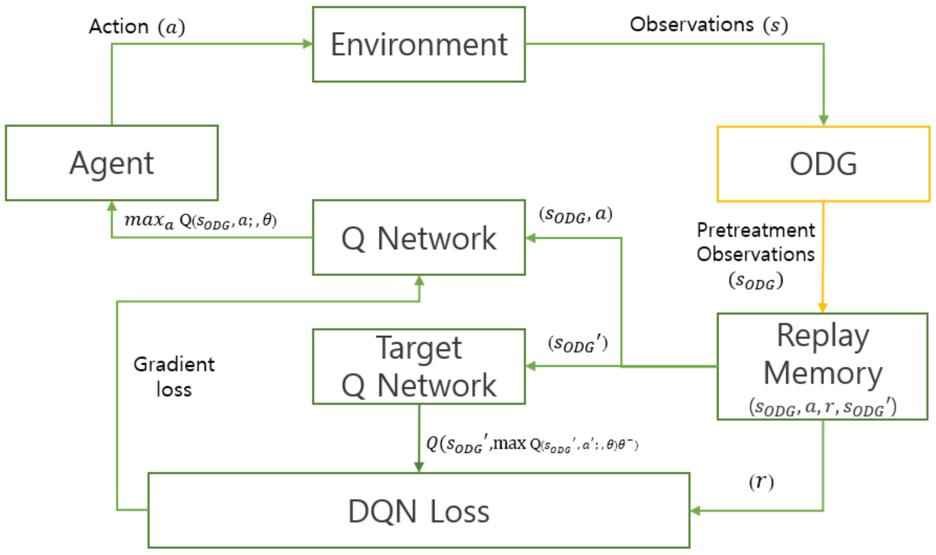

Action (a)
Environment
Observations (s)
Agent
ODG
max_a Q$(s_{ODG}, a;, \theta)$
Q Network
(s_{ODG}, a)
Pretreatment Observations (s_{ODG})
Gradient loss
Target Q Network
(s_{ODG}')
Replay Memory $(s_{ODG}, a, r, s_{ODG}')$
$Q(s_{ODG}'$,max Q$(s_{ODG}', a';, \theta)\theta^-)$
(r)
DQN Loss

Figure 2. ODG DQN structure.

LiDAR information, a principal component in autonomous driving systems, is preprocessed via the ODG module, subsequently offering the processed data to the RL approach as the state value. Through the use of a Gaussian distribution, the ODG module converts LiDAR information into continuous values. As depicted in Figure 3, the creation of a unique state happens when an agent selects an action, preventing the duplication of action values and facilitating a more efficient selection of the optimal action value in accordance with the equation.

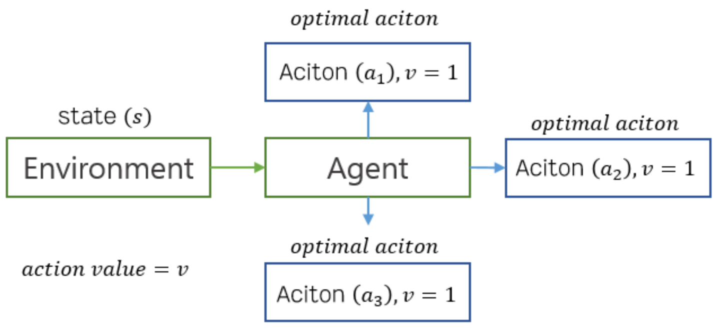


Figure 3. Overestimation in Q-learning.

For the implementation of our proposed algorithm to RL using LiDAR information, a standard procedure in autonomous vehicles, we employ ODG-based preprocessed LiDAR information. As demonstrated in Figure 4, the yellow line corresponds to the original LiDAR data, whereas the blue line symbolizes post-processed data. These data include information on obstacle location and size, derived using Equation (5) with ODG [19].

$$a = f_{rep}(\theta_i) = \sum_{k=1}^{n} A_k \exp\left(-\frac{(\theta_k - \theta_i)^2}{2\sigma_k^2}\right), \tag{5}$$

where

$$A_k = (d_{\max} - d_k)\exp(0.5), \tag{6}$$

$$Q(s_t^{\Delta}, a_t) = \mathbb{E}[r_t + \gamma max_a Q(s_{t+1}^{\Delta}, a)]. \tag{7}$$

In contrast to the overlapping LiDAR information provided by conventional methods, ODG supplies non-overlapping LiDAR data, adjusting the maximum range according to the obstacle's size and distance. This preprocessing enables the agent to make more efficient decisions related to optimal action values based on the processed information, thereby enhancing both the speed and efficiency of learning. The reward function used for training is defined in Equation (8).

$$R = R_g + R_v + R_\psi. \tag{8}$$

where R_g represents the target reward, R_v denotes the reward for speed, and R_ψ signifies the reward for steering angle.

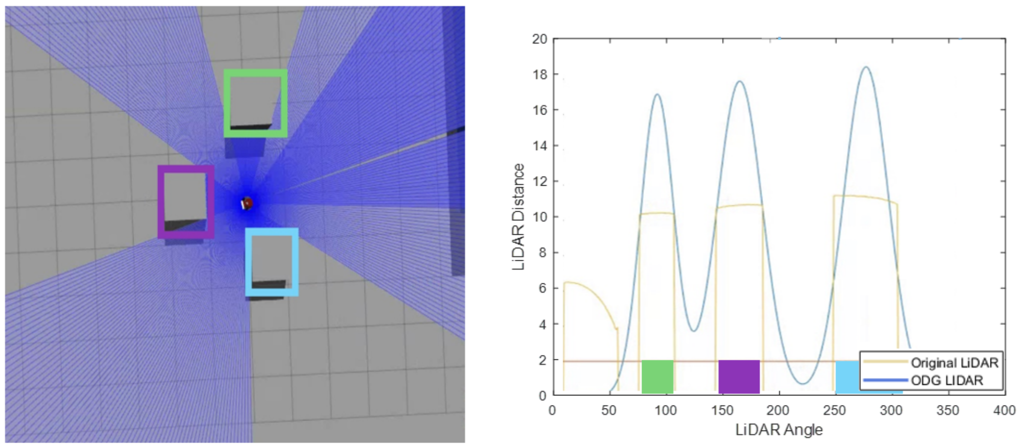


Figure 4. Difference between traditional LiDAR and ODG information.

2.2. Guide ODG DQN

The soft actor critic (SAC) method [29] is a robust approach that allows for the observation of multiple optimal values while avoiding the selection of impractical paths. This facilitates a more extensive policy exploration. The SAC employs an efficient and stable entropy framework for the continuous state and action space. As delineated in Equation (10) with SAC [29], the SAC learns the optimal Q function through updating Q-learning via the maximum entropy RL method.

$$\sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t)], \tag{9}$$

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))]. \tag{10}$$

The algorithm initially makes the guide value sparse and, as learning progresses, gradually densifies it, employing the gamma value as outlined in Equation (12) with SAC [29]. The term min A is representative of the environmental vehicle.

$$min_\psi A < \Delta_\psi < max_\psi A, \tag{11}$$

$$v = max_\psi A - |\Delta_\psi|. \tag{12}$$

A report on hierarchical deep RL, an approach that implements RL via multiple objectives, emphasized the need to solve sparse reward problems as environments become increasingly diverse and complex. Normally, in problems tackled by RL, rewards are generated for each state, like survival time or score. Every state is linked to an action, receives a reward, and identifies the Q-value so as to maximize the sum of the rewards. However, there are instances where a reward may not be received for each state. These scenarios are referred to as sparse rewards.

$$Q(s_t,a_t) = \mathbb{E}[r_t + \alpha\gamma\mathcal{G}max_a Q(s_{t+1},a) + (1-\alpha)\gamma max_a Q(s_{t+1},a)], \tag{13}$$

where

$$\mathcal{G}max_a f(a) := Guide_{action}(S^{\Delta}_{t+1}). \tag{14}$$

Our proposed solution to these issues is the guide-ODG-DNQ model that integrates SAC with ODG-DQN. This proposed guide-ODG-DQN algorithm transforms the initial Q-value from the state value. This value is extracted from the environment, and it is connected with the ODG formula, which is our prior knowledge, and the LiDAR value extracted with ODG, as depicted in Figure 5. The algorithm extracts a guide action that minimizes the cases where a reward is not received for every state.

The guide-ODG-DQN is designed to store high-quality information values in the replay memory from the outset based on prior knowledge. The agent then continues learning based on this prior knowledge, facilitating easier adaptation to various environments and enabling faster and more stable convergence. Moreover, to prevent over-reliance on prior knowledge that could compromise the effectiveness of RL, the agent learns from its own experiences during the learning process, which are represented by the gamma value. The agent also contrasts this newly learned information with the values derived from the existing prior knowledge. Consequently, our proposed guide-ODG-DQN mitigates the sparse reward phenomenon, thereby enhancing both the stability and efficiency of the learning process.

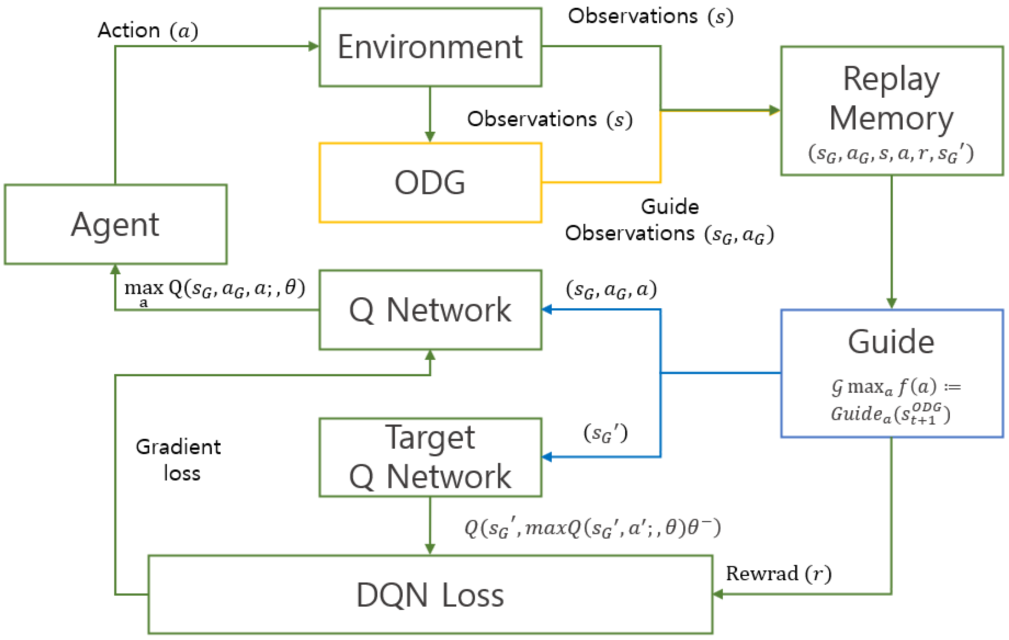

Action (a)
Environment
Observations (s)
Replay Memory
$(s_G, a_G, s, a, r, s_G')$
Observations (s)
ODG
Agent
Guide
Observations (s_G, a_G)
$\max_a Q(s_G, a_G, a; , \theta)$
Q Network
(s_G, a_G, a)
Guide
$\mathcal{G}\max_a f(a) \coloneqq Guide_a(s_{t+1}^{ODG})$
Gradient loss
Target Q Network
(s_G')
$Q(s_G', maxQ(s_G', a'; , \theta)\theta^-)$
Rewrad (r)
DQN Loss

Figure 5. Structure of guide-ODG-DQN.

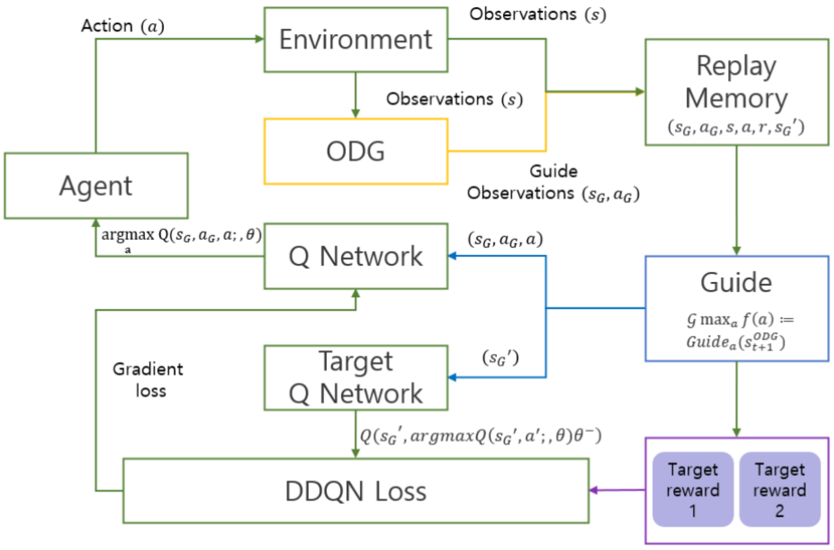
2.3. *Meta-Learning-Based Guide ODG DDQN*

RL is fundamentally a process of learning through trial and error. The RL agent must experience a diverse set of situations, making decisions in each scenario to understand which actions yield the highest rewards. Striking a balance between experimentation, to ensure no high-reward actions are overlooked, and leveraging acquired knowledge to maximize rewards is crucial. However, achieving this balance typically necessitates numerous trials and, consequently, large volumes of data. Training an RL agent with excessive data might result in overfitting, wherein the agent conforms too closely to the training data and fails to generalize well to new circumstances.

To overcome these limitations, we introduce a novel method known as meta-learning-based guide-ODG-DDQN. This approach involves storing rewards for each step an integral part of RL in the replay memory, with the stored rewards divided according to the number of targets to be learned as shown in Figure 6. This model facilitates few-shot learning within RL by training the model to recognize similarities and differences, thus preparing it to perform proficiently in unfamiliar environments with minimal data. The training is guided by two main objectives. The first is to train the target model using the initial reward, while the second is to continue learning by reducing the weight assigned to the initial reward and increasing the weight of the reward for the subsequent target, as depicted in Equation (15).

$$R_\Lambda = \gamma R_{J_1} + (1-\gamma) R_{J_2}, \gamma \in [0,1]. \tag{15}$$

By applying our meta-learning-based ODG RL, the model achieves multiple significant outcomes. It allows for the training of a universal model that can operate reliably across various environments. The model's efficiency of learning is boosted due to its ability to identify similarities and differences. Furthermore, learning can proceed using a common target while preserving the existing target. In essence, the proposed algorithms augment the efficiency and stability of traditional RL methods, safely accelerating the learning speed within a virtual environment, which ultimately improves efficiency when the model is implemented in real-world environments.



Action (a)
Environment
Observations (s)
Replay Memory
$(s_G, a_G, s, a, r, s_G{}')$
Observations (s)
ODG
Agent
Guide
Observations (s_G, a_G)
$\underset{a}{\text{argmax}}\, Q(s_G, a_G, a;, \theta)$
Q Network
(s_G, a_G, a)
Guide
$\mathcal{G}\max_a f(a) := Guide_a(s_{t+1}^{ODG})$
Gradient loss
Target Q Network
$(s_G{}')$
$Q(s_G{}', argmaxQ(s_G{}', a';, \theta)\theta^-)$
DDQN Loss
Target reward 1
Target reward 2

Figure 6. Structure of meta-learning-based guide-ODG-DDQN.

3. Experiment

In the process of validating our proposed algorithm, we conducted an experiment evaluating key aspects such as learning efficiency, stability, strength, and adaptability to complex environments. Learning efficiency was determined by examining the highest reward achieved as learning started to converge, in relation to the number of frames experienced in the virtual environment. The DQN algorithm was used as the basis to analyze the rate of convergence and the magnitude of the reward. For the evaluation of learning stability, we assessed the consistency between the path plan generated through RL (P_{RL}) and the target path produced by ODG (P_{ODG}). Here, P_k represents the set of paths. This assessment involved the use of the root mean square error (RMSE), where P_{RL} and P_i represent the path plans formed through RL and ODG, respectively. The route yielding the highest reward was considered optimal. Finally, we evaluated the algorithm's performance in complex environments. This part of the evaluation was focused on the vehicle's ability to effectively navigate through real world maps, leveraging learning strength. We also tested the resilience and adaptability of the algorithm when faced with unfamiliar scenarios without further training. Metrics such as entry and exit speed, as well as racing track lap time, were used to measure performance. The evaluation environments were chosen with care for distinct aspects of the study: the Gazebo map was used to evaluate learning efficiency and stability, the Sochi map for learning strength, and the Silverstone map to test adaptability to complex conditions, as shown in Figure 7. The experiment setup was designed to reflect real world dimensions, such that each unit length in the simulation corresponded to one meter in reality [30,31].

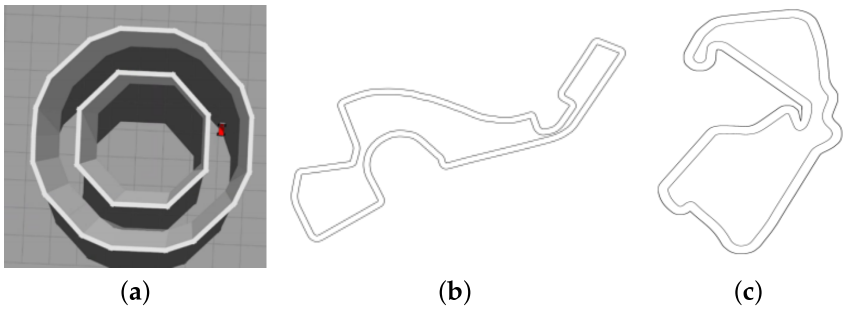


Figure 7. Map environment. (**a**) Gazebo map; (**b**) Sochi map; and (**c**) SILVERSTONE map.

First, the index for learning efficiency is determined as follows. As learning begins to converge, the learning efficiency corresponding to the highest reward for the number of frames (in millions) experienced in the virtual environment is considered. Based on the DQN algorithm, we evaluate how fast convergence occurs and how high the reward is.

Second, the evaluation metric for learning stability assesses how well the path plan generated through RL matches the target path pursued. The path plan created by RL in the virtual environment, P_{RL}, and the path plan created with the ODG, P_{ODG}, are represented in terms of the $RMSE$. Both P_{ODG} and P_{RL} are individually compared with the reference path, and their respective errors are calculated using Equation (16).

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}, \tag{16}$$

where $\hat{y}_i$ represents the path generated by ODG in P_{ODG}, which is known to exhibit high real-time performance and stability, and y_i corresponds to P_{RL}, which is the path plan generated through RL. The optimal route with the highest reward is considered. n is the number of steps the agent operates in the simulation environment, corresponding to the episodic steps in RL. A smaller $RMSE$ corresponds to a more stable.

Finally, we evaluate the performance in complex environments, as depicted in Figure 7. The assessment metrics focus on how effectively the vehicle navigates through intricate obstacles while ensuring safety and speed. We showcase the learning strength in the Sochi Circuit and the learning diversity in the Silverstone Circuit. For this evaluation, we utilized real maps and employed the metrics of "Enter and Exit Speed" and "Racing Track Lap Time" to assess the agent's performance. In summary, our results demonstrate the learning strength and diversity of the proposed algorithm in handling complex environments and showcase its robustness when encountering new scenarios without further training.

3.1. Learning Performance and Efficiency Evaluation

The hyperparameters used in set up are listed in Table 1. The set up is aimed at verifying the efficiency of the algorithm to be applied in a real environment. Therefore, reducing the learning time is the priority. To evaluate whether learning efficiency and stability are ensure, a basic circular map is selected, and a performance comparison experiment is conducted for each RL algorithm: DQN, ODG-DQN, DDQN, and guide-ODG-DQN. The agent model and environment used in the experiment are shown in Figure 8.

Table 1. Hyperparameters in set up.

	Hyper Parameters
v	car speed $\in [0, 0.7]$ m/s
a	action (steering angle) $\in [-1/3,\ 1/3]^\circ$
ψ	angular speed = $\in [0, 0.1]$ m/s
τ	update target network = 10,000
α	learning Rate = 0.00025
M_{size}	minibatch Size = 64
γ	discount = 0.99
Eps	exploration Rete = 1
R_v	$v - a * v$
R_ψ	$5 = argmax_\theta d_L$, $0 = otherwise$
R_g	crash = -200, finish = 300, checkpoint = 100

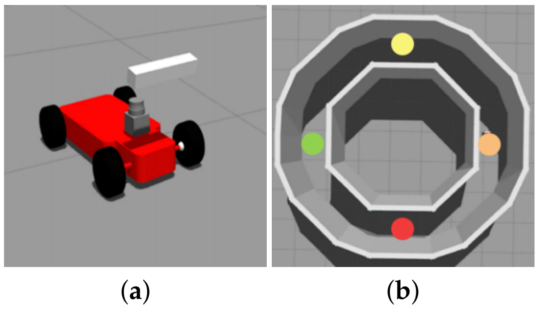


Figure 8. Set up check point and arrival point. End point (red), point1 (orange), point2 (yellow), and point3 (green). (**a**) Agent; (**b**) Check point.

The reward function used for training is defined in Table 1. First, to compare the DQN and ODG-DQN algorithms for the Gazebo map, we determine the number of steps in which the checkpoint is reached during training, as indicated in Table 2. In DQN, over 50% of untrained failure cases are overestimated, whereas in ODG-DQN, 10% of untrained failure cases occur, corresponding to overestimation occurrence reduced by 80%.

Table 2. Epochs of algorithm passing checkpoints the first time. *Fail*: *Overestimation*.

Experiment	Algorithm (Step)							
	DQN				ODG-DQN			
	P.1	**P.2**	**P.3**	**P.4**	**P.1**	**P.2**	**P.3**	**P.4**
No. 1	5	10	15	20	4	7	12	18
No. 2	6	14	Fail	Fail	4	9	13	17
No. 3	7	11	16	20	4	7	12	16
No. 4	Fail	Fail	Fail	Fail	4	8	12	17
No. 5	Fail	Fail	Fail	Fail	5	8	12	18
No. 6	6	12	16	20	5	8	12	17
No. 7	5	10	16	19	4	8	13	17
No. 8	5	13	15	19	4	9	12	17
No. 9	Fail	Fail	Fail	Fail	4	8	Fail	Fail
No. 10	Fail	Fail	Fail	Fail	5	9	12	17
Average	5.75	13.3	15.6	19.6	4.3	8.9	12.2	17.1

Guide-ODG-DQN and DDQN are compared under the same conditions. Figure 9a shows the results of learning in terms of the epoch values of the safe convergence section for each algorithm implemented 10 times. As indicated in Table 3, the learning convergence rate increases by 51.7%, 89%, and 16.8%, respectively, compared with the other algorithm. Figure 9b shows that the learning is inappropriate due to overestimation in the case of DQN. In the cases of ODG-DQN, DDQN, and guide-ODG-DQN, learning converges at approximately 500, 300, and 200 epochs, respectively the results are summarized in Table 4.

Table 3. Decrease in the epochs of ODG-DQN.

Algorithm	Decrease in the Epochs
DQN $\rightarrow$ ODG-DQN	51.7%
DQN $\rightarrow$ guide-ODG-DQN	89%
DDQN $\rightarrow$ guide-ODG-DQN	16.8%

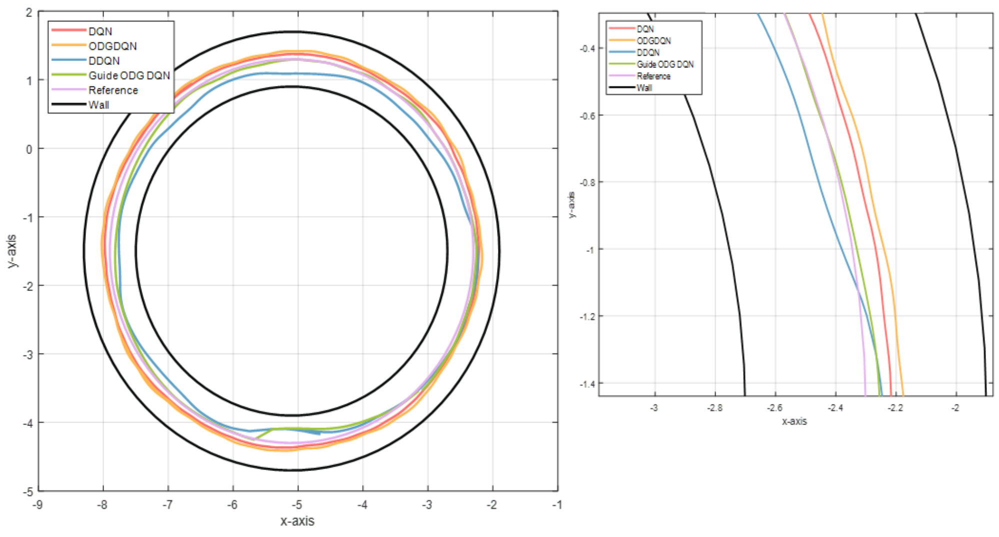
Next, to evaluate the stability of the RL results, the center line corresponding to the Gazebo map is applied as a reference. The path generated by each algorithm is shown in Figure 9a. Moreover, Table 5 shows the results obtained by comparing the algorithms in terms of the RMSE, as defined in Equation (16). The RMSE for guide-ODG-DQN is 0.04, corresponding to the highest stability. The guide-ODG-DQN achieves the lowest RMSE, corresponding to the highest stability, as shown in Figure 9c.

Table 4. Summary of normalized performance up to 10 cycles of play on track. *Fail*: *Overestimation*.

Experiment	**Algorithm (Epoch)**			
	DQN	**ODG DQN**	**DDQN**	**Guide ODG DQN**
No. 1	1342	587	181	134
No. 2	Fail	621	175	175
No. 3	1416	576	177	121
No. 4	Fail	572	201	143
No. 5	Fail	610	182	172
No. 6	1321	593	177	144
No. 7	1422	631	188	155
No. 8	1452	579	192	177
No. 9	Fail	Fail	181	165
No. 10	Fail	668	185	143
Average Value	1391	672	184	153

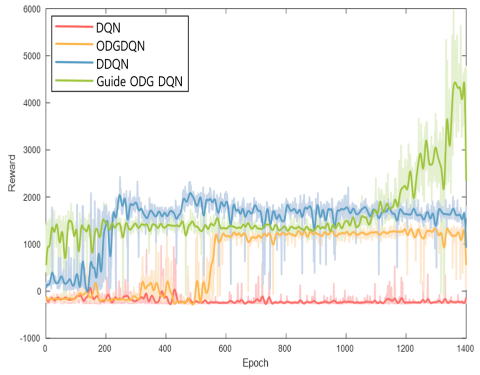
Table 5. RL RMSE.

Algorithm	**RMSE**
DQN	0.0745
ODG-DQN	0.1142
DDQN	0.1082
Guide-ODG-DQN	0.0395

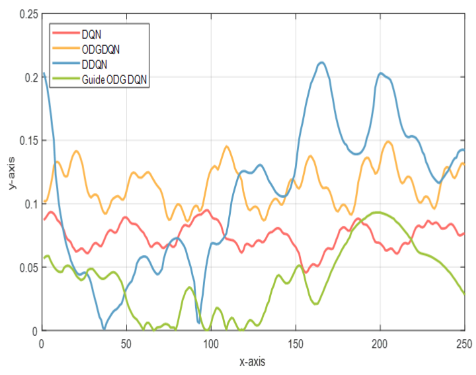


(**a**)

Figure 9. *Cont.*



(**b**)

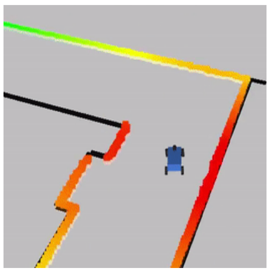


(**c**)

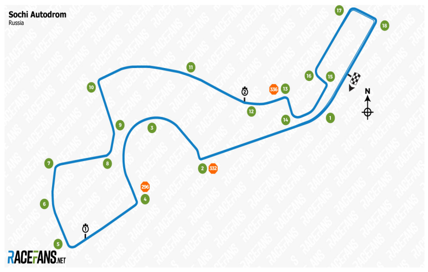
Figure 9. Experiment set up result. (**a**) RL algorithm path comparison; (**b**) RL reward graph; and (**c**) RL RMS.

3.2. Results through Simulation that Mimics the Real Environment

The hyperparameters values used in circuit are listed in Table 6. As the evaluation metric for a complex environment, shown in Figure 10a, the method of learning the speed is considered instead of that for learning angles. Therefore, the angle is set to that associated with the ODG to ensure stability. The reward function used for all RL frameworks is the same as that defined in Table 6. Figure 10b shows the official competition map provided by F1TENTH. Using the control point specified in the actual Sochi Autodrom map, we compare the path in the winding road and hairpin curve.



(**a**)



(**b**)

Figure 10. Actual existing Sochi Autodrom map information, officially provided by F1TENTH. (**a**) Agent; (**b**) control point.

Table 6. Hyperparameters for the F1TENTH.

	Hyperparameters
a	action (car speed) $\in [0, 20]$ m/s
Δ_ψ	ODG steering angle $\in [-12,\ 12]^\circ$
max_v	max car speed = 20 m/s
τ	update target network = 10,000
α	learning rate = 0.00001
M_{size}	minibatch size = 128
γ	discount = 0.99
Eps	exploration rate = 1
R_Λ	$a - max_v - \lvert\Delta_\psi\rvert$
R_ψ	$100 = argmax_\theta d_L$, $0 = otherwise$
R_g	crash = -100, finish = 200, and episode step = $episode_\Lambda$

The agent starts at the wall of control point 1. Linear velocity graphs for ODG, Gap Follower, DDQN, and meta ODG DDQN are shown in Figure 13. In this case, 100 points on the x-axis are used as control points, and 100-step linear velocity values are output on both sides based on these values.

3.2.1. Sochi International Street Circuit

The Sochi Autodrom, previously known as the Sochi International Street Circuit and the Sochi Olympic Park Circuit, is a 5.848 km permanent race track in the settlement of Sirius next to the Black Sea resort town of Sochi in Krasnodar Krai, Russia, as shown in Figure 11. Here, the learning strength is demonstrated, in the Sochi Circuit.

(**a**) (**b**)

Figure 11. Circuit. (**a**) Sochi International Street Circuit; (**b**) Silverstone Circuit.

Table 7 lists the average speed for each control point for each algorithm. Table 7 shows that the ODG algorithm that prioritizes stability achieves the lowest value of 7.66, and the meta ODG DDQN achieves the highest value of 8.58. In other words, the meta ODG DDQN completes the Sochi Autodrom with a speed 12.01% higher than that of the ODG.

Table 7. Average speed control point.

Method	**Algorithm**			
	ODG	**Gap Follower**	**DDQN**	**Meta ODG DDQN**
No. 1	8.98	7.98	8.23	8.48
No. 2	8.10	7.79	7.93	8.69
No. 3	8.04	8.02	**7.78**	**8.86**
No. 4	7.74	7.75	8.26	8.60
No. 5	7.84	7.79	7.94	8.49
No. 6	8.41	7.95	8.59	8.66
No. 7	7.58	7.76	8.35	8.59
No. 8	7.71	7.72	8.24	**8.25**
No. 9	7.41	7.67	7.81	8.34
No. 10	7.82	7.72	8.17	8.38
No. 11	**9.15**	8.03	**8.66**	8.82
No. 12	9.08	**8.04**	8.23	8.48
No. 13	7.40	7.80	8.53	8.51
No. 14	**5.87**	7.45	8.08	8.73
No. 15	6.89	7.64	8.05	8.70
No. 16	5.44	**7.30**	7.88	8.30
No. 17	7.99	7.69	8.36	8.86
No. 18	6.34	7.42	8.12	8.61
Average speed	**7.66**	**7.75**	**8.17**	**8.58**

As shown in Figure 12a, to examine the speeds of entry and exit at the control point, which are of significance in a racing game, the entry and exit speed for each algorithm are presented in Table 8. In the case of ODG, which is an algorithm that prioritizes stability, as shown in Figure 12b, understeer or oversteer does not occur [32,33]. A report on racing high-performance tires [34] indicates that in this driving method, the vehicle enters at a high speed and exits at a low speed.

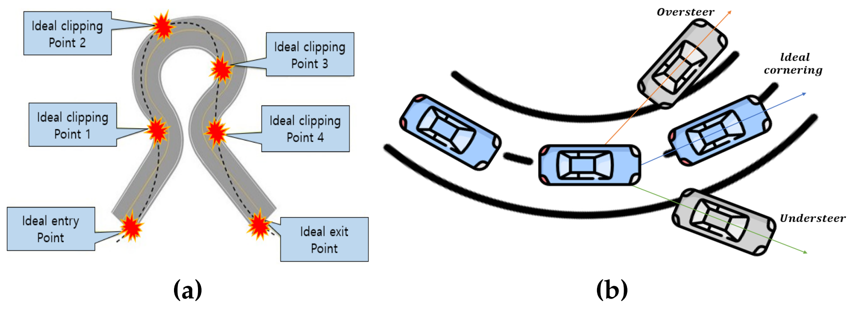


Figure 12. Corner driving: (**a**) clipping point; and (**b**) understeer and oversteer.

As shown in Table 9, ODG selects a drive with a 13.9% speed reduction. The racing algorithm, Gap Follower, uses an out-in-out driving method with a 3.41% deceleration. However, the DDQN and meta ODG DDQN algorithms lead to oversteer to achieve maximum speed based on the angle extracted from the ODG, which pursues stability, causing the vehicle to spin inward compared to the expected route. So, DDQN and meta ODG DDQN show a driving method without deceleration at control points of 1.33% and 0.46%, respectively, by drawing a path.

Table 8. Control point enter and exit speed.

	Enter and Exit Speed (m/s)							
Method	**ODG**		**GAP Follower**		**DDQN**		**Meta ODG DDQN**	
Control Point	**Enter**	**Exit**	**Enter**	**Exit**	**Enter**	**Exit**	**Enter**	**Exit**
No.1	9.46	8.59	7.99	8.04	8.07	8.46	8.42	8.63
No.2	**9.28**	**6.98**	8.07	7.58	8.02	7.91	8.86	8.61
No.3	8.08	8.08	8.05	8.06	7.63	8.00	8.83	8.97
No.4	8.69	6.86	**8.07**	**7.49**	8.16	8.44	8.75	8.54
No.5	8.83	6.92	7.98	7.66	8.21	7.74	**8.54**	**8.51**
No.6	8.59	8.32	7.95	8.03	**8.61**	**8.65**	8.77	8.64
No.7	8.42	6.81	8.05	7.54	8.26	8.51	**8.83**	**8.45**
No.8	8.35	7.15	7.88	7.62	**8.63**	**7.91**	8.25	8.34
No.9	7.11	7.77	7.56	7.86	8.01	7.68	8.24	8.54
No.10	8.97	6.74	8.03	7.49	8.38	8.03	8.34	8.52
No.11	**9.14**	**9.25**	8.06	8.07	8.56	8.84	8.96	8.77
No.12	9.23	9.02	**8.07**	**8.07**	8.35	8.18	8.64	8.41
No.13	8.51	6.37	8.07	7.60	8.70	8.43	8.39	8.72
No.14	6.33	5.48	7.61	7.36	8.43	7.80	8.71	8.85
No.15	7.63	6.22	7.89	7.47	8.13	8.04	8.85	8.65
No.16	5.84	5.07	7.48	7.17	8.05	7.78	8.58	8.10
No.17	8.96	7.09	8.00	7.44	8.42	8.38	9.02	8.79
No.18	7.36	5.39	7.67	7.23	8.18	8.12	8.59	8.72
Average speed	**8.27**	**7.12**	**7.92**	**7.65**	**8.27**	**8.16**	**8.64**	**8.60**

Moreover, the lap time is compared for the map shown in Figure 10b by considering two laps (based on the F1TENTH formula). Table 10 indicates that meta ODG DDQN achieves the highest speed.

Linear velocity graphs for ODG, Gap Follower, DDQN, and meta ODG DDQN are shown in Figure 13; using the control point specified in the actual Sochi Circuit, we compare the path in the winding road and hairpin curve, as shown in Figure 14.

Table 9. Enter and exit speed.

Algorithm	Speed Reduction
ODG	13.9%
Gap Follower	3.41%
DDQN	1.33 %
Meta ODG DDQN	0.46%

Table 10. Racing track lap time.

Algorithm	Racing Track 2 Lap Time (s)
ODG	117.09
Gap Follower	115.68
DDQN	115.41
Meta ODG DDQN(ours3)	109.85

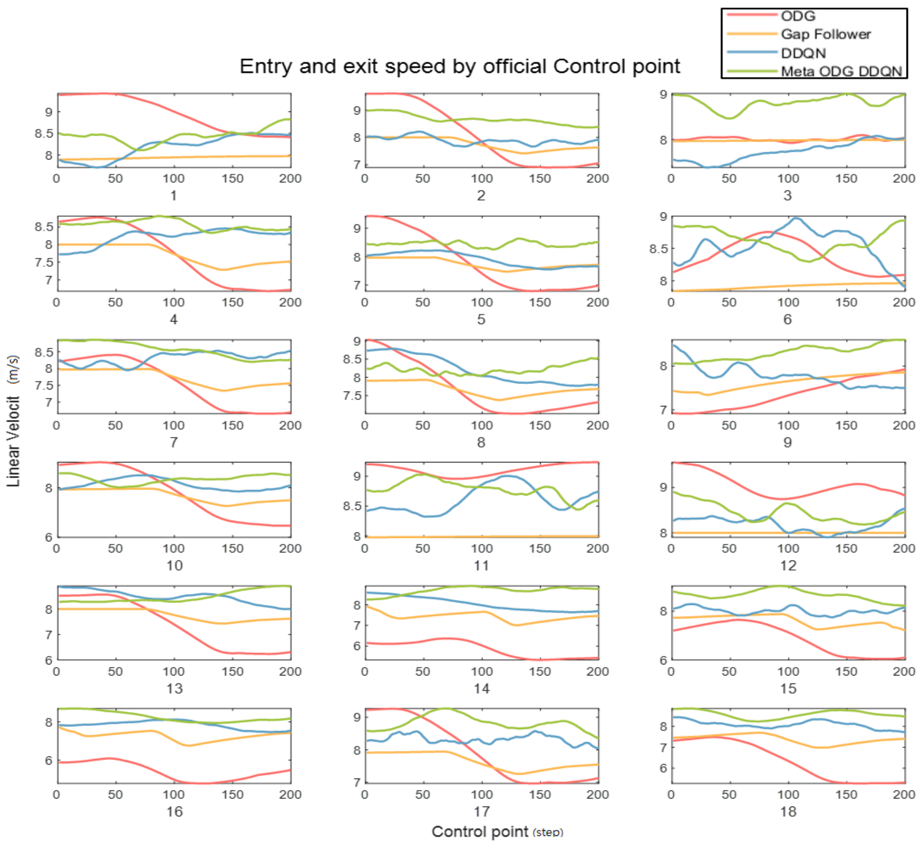


Figure 13. Control point speed.

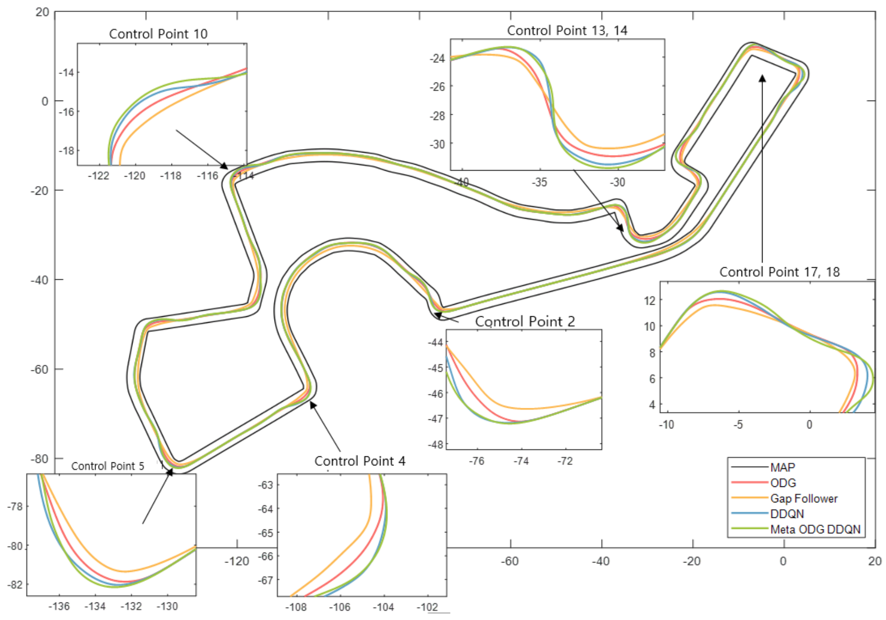


Figure 14. Map of the Sochi path.

3.2.2. Silverstone Circuit

Silverstone Circuit is a motor racing circuit in England, near the Northamptonshire villages of Towcester, Silverstone, and Whittlebury, as shown in Figure 11. In this result, the learning diversity in the Silverstone Circuit is demonstrated.

Using the RL model trained in Map Sochi, we conduct an experiment to determine the degree of robustness to unfamiliar and complex environments. Therefore, we use the algorithms ODG, Gap Follower, DDQN, and meta ODG DDQN. In addition, the meta ODG DDQN algorithm is trained in a new environment. In other words, the robustness of the new environment (c) was compared based on the driving style learned in (b) shown in Figure 7.

Table 11 presents the results for a new environment. DDQN fails; however, meta ODG DDQN exhibits high performance with the lowest lap time, as shown in Figure 15. In this result, the learning diversity in the Silverstone Circuit is demonstrated.

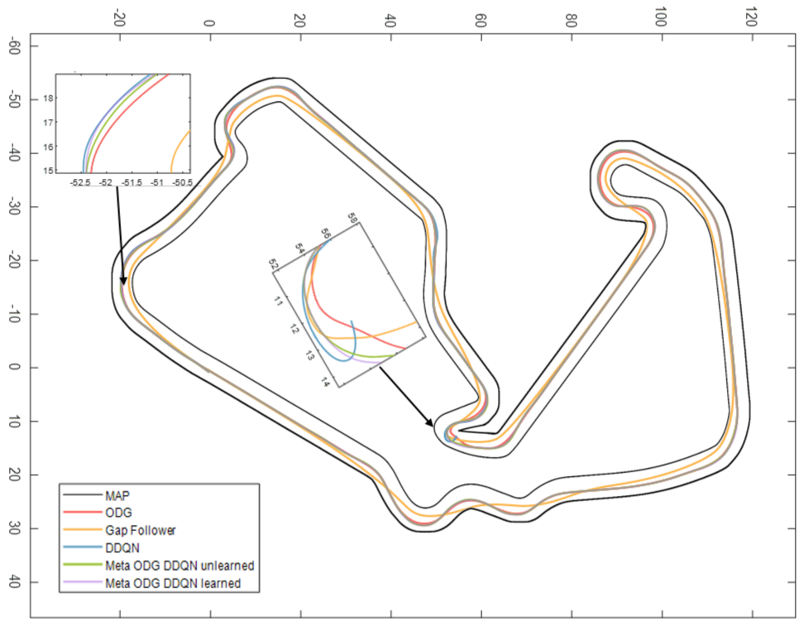


Figure 15. Map (b) SILVERSTONE path.

Table 11. Map (b) SILVERSTONE racing track lap time (two laps). *Fail* = Crash.

Algorithm	Lap Time (s)
ODG	117.39
Gap follower	110.99
DDQN	*Fail*
Meta ODG DDQN unlearned	108.60
Meta ODG DDQN learned	108.43

4. Conclusions

This paper introduces a novel RL-based autonomous driving system technology that implements ODG, SAC, and meta-learning algorithms. In autonomous driving technology, perception, decision-making, and control processes intertwine and interact. This work addresses the issues of the overestimation phenomenon and sparse rewards problems by applying the concept of prior knowledge. Furthermore, the fusion of meta-learning-based RL yields robust results in previously untrained environments.

The proposed algorithm was tested on official F1 circuits, a racing simulation with complex dynamics. The results of these simulations emphasize the exceptional performance of our method, which exhibits a learning speed up to 89% faster than existing algorithms in these environments. Within the racing context, the disparity between entry and exit speeds is a mere 0.46%, indicating the smallest reduction ratio. Moreover, the average driving speed was found to be up to 12.01% higher.

The primary contributions of this paper comprise a unique combination addressing the challenges of overestimation phenomenon and sparse rewards problems effectively in RL. Another major contribution is the demonstrated robust performance of the integrated meta-learning-based RL in previously untrained environments, thereby showcasing its adaptability and stability. Furthermore, we validated the performance of our proposed method via complex racing simulations, particularly on official F1 circuits. The results highlighted its superior performance in terms of learning efficiency, speed, stability, and adaptability.

In essence, this paper tackles the significant challenges encountered during the reinforcement learning process by introducing an algorithm that bolsters the efficiency and stability of RL. The high-fidelity simulations used in this study offer a realistic testing environment closely mirroring real-world conditions. Given these advancements, our proposed algorithm demonstrates significant potential for real-world applications, particularly in autonomous vehicles where learning efficiency and operational stability are of the utmost importance.

As for future research, we suggest adding various multi-tasks to verify stable and efficient learning in more complex environments. Based on this, we aim to study efficient RLs in real environments through meta-learning, with as few iterations as possible.

Author Contributions: Conceptualization, S.-H.J. and M.-T.L.; formal analysis, S.-H.J., W.-J.A. and M.-T.L.; methodology, S.-H.J., Y.-J.K., M.-T.L. and D.-S.P.; software, S.-H.J., H.-G.H. and D.-S.P.; validation, M.-T.L. and D.-S.P.; writing—original draft, S.-H.J. and M.-T.L.; and writing—review and editing, M.-T.L. and D.-S.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) (grant no. NRF-2022R1F1A1073543).

Data Availability Statement: The code used in this paper is attached to the following Github address: https://github.com/jang1229/Gazebo-ODG-DQN; https://github.com/jang1229/F1Thenth-ODGPF.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Berecz, C.E.; Gabor, K. Dangers in autonomous vehicles. In Proceedings of the 2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary, 21–22 November 2018.
2. Hoel, C.J.; Wolff, K.; Laine, L. Automated speed and lane change decision making using deep reinforcement learning. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 2148–2155.
3. Qiao, Z.; Muelling, K.; Dolan, J.M.; Palanisamy, P.; Mudalige, P. Automatically generated curriculum based reinforcement learning for autonomous vehicles in urban environment. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 1233–1238.
4. Barreto, A.; Hou, S.; Borsa, D.; Silver, D.; Precup, D. Fast reinforcement learning with generalized policy updates. *Proc. Natl. Acad. Sci. USA* **2020**, *117*, 30079–30087. [CrossRef] [PubMed]
5. Bellman, R. A Markovian decision process. *J. Math. Mech.* **1957**, *6*, 679–684. [CrossRef]
6. Lecun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
7. Peng, B.; Sun, Q.; Li, S.E.; Kum, D.; Yin, Y.; Wei, J.; Gu, T. End-to-end autonomous driving through dueling double deep Q-network. *Automot. Innov.* **2021**, *4*, 328–337. [CrossRef]
8. Yang, Y.; Pan, Y.; Xu, C.; Wunsch Donald, C. Hamiltonian-driven adaptive dynamic programming with efficient experience replay. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–13. [CrossRef]
9. Sutton, R.S.; Barto Andrew, G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
10. Gangopadhyay, B.; Soora, H.; Dasgupta, P. Hierarchical program-triggered reinforcement learning agents for automated driving. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 10902–10911. [CrossRef]
11. Dayal, A.; Cenkeramaddi, L.R.; Jha, A. Reward criteria impact on the performance of reinforcement learning agent for autonomous navigation. *Appl. Soft Comput.* **2022**, *126*, 109241. [CrossRef]
12. Watkins, C.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
13. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. *Proc. Aaai Conf. Artif. Intell.* **2016**, *30*. [CrossRef]
14. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
15. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. *Int. Conf. Mach. Learn.* **2016**, *48*, 1995–2003.
16. Burrell, J. How the machine 'thinks': Understanding opacity in machine learning algorithms. *Big Data Soc.* **2016**, *3*, 2053951715622512. [CrossRef]
17. Bai, C.; Wang, L.; Wang, Y.; Wang, Z.; Zhao, R.; Bai, C.; Liu, P. Addressing hindsight bias in multigoal reinforcement learning. *IEEE Trans. Cybern.* **2021**, *53*, 392–405. [CrossRef] [PubMed]
18. Kulkarni, T.D.; Narasimhan, K.; Saeedi, A.; Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 3675–3683.
19. Cho, J.H.; Pae, D.S.; Lim, M.T.; Kang, T.K. A real-time obstacle avoidance method for autonomous vehicles using an obstacle dependent Gaussian potential field. *J. Adv. Transp.* **2018**, *2018*, 5041401. [CrossRef]
20. Pae, D.S.; Kin, G.H.; Kang, T.K.; Lim, M.T. Path Planning Based on Obstacle-Dependent Gaussian Model Predictive Control for Autonomous Driving. *Appl. Sci.* **2021**, *11*, 3703. [CrossRef]
21. Kim, J.C.; Pae, D.S.; Lim, M.T. Obstacle Avoidance Path Planning based on Output Constrained Model Predictive Control. *Int. J. Control. Autom. Syst.* **2019**, *17*, 2850–2861. [CrossRef]
22. Botvinick, M.; Ritter, S.; Wang, J.X.; Kurth-Nelson, Z.; Blundell, C.; Hassabis, D. Reinforcement learning, fast and slow. *Trends Cogn. Sci.* **2019**, *23*, 408–422. [CrossRef]
23. Sallab, A.; Abdou, M.; Perot, E.; Yogamani, S. Deep reinforcement learning framework for autonomous driving. *Electron. Imaging* **2017**, *2017*, 70–76. [CrossRef]
24. Korah, T.; Medasani, S.; Owechko, Y. Strip histogram grid for efficient lidar segmentation from urban environments. In Proceedings of the Computer Vision and Pattern Recognition 2011 Workshops, Colorado Springs, CO, USA, 20–25 June 2011; pp. 74–81.
25. Ramachandran, D.; Amir, E. Bayesian Inverse Reinforcement Learning. *Int. Jt. Conf. Artif. Intell.* **2007**, *7*, 2586–2591.
26. Cetin, E.; Oya, C. Learning Pessimism for Reinforcement Learning. *Proc. Aaai Conf. Artif. Intell.* **2023**, *37*, 6971–6979. [CrossRef]
27. Menache, I.; Shie, M.; Nahum, S. Basis function adaptation in temporal difference reinforcement learning. *Ann. Oper. Res.* **2005**, *134*, 215–238. [CrossRef]
28. Meng, L.; Rob, G.; Dana, K. The effect of multi-step methods on overestimation in deep reinforcement learning. In Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 10–15 January 2021.
29. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *Int. Conf. Mach. Learn.* **2018**, *80*, 1861–1870.
30. Ueter, N.; Chen, K.; Chen, J. TProject-based CPS education: A case study of an autonomous driving student project. *IEEE Des. Test* **2020**, *37*, 39–46. [CrossRef]

31. Betz, J.; Zheng, H.; Liniger, A.; Rosolia, U.; Karle, P.; Behl, M.; Krovi, V.; Mangharam, R. Autonomous vehicles on the edge: A survey on autonomous vehicle racing. *IEEE Open J. Intell. Transp. Syst.* **2022**, *3*, 458–488. [CrossRef]
32. Hosseinian, A.A.; Melzi, S. Numerical analysis of the influence of an actively controlled spoiler on the handling of a sports car. *J. Vib. Control* **2018**, *24*, 5437–5448. [CrossRef]
33. Nguyen, T.A. Establishing the Dynamics Model of the Vehicle Using the 4-Wheels Steering Systems. *Math. Model. Eng. Probl.* **2020**, *7*, 436–440. [CrossRef]
34. Paul, H. *The Racing High-Performance Tire: Using Tires to Tune for Grip Balance (R-351)*; Society of Automotive Engineers Inc.: Warrendale, PA, USA, 2003; pp. 60–61.