

Article

# Autonomous Driving Control Based on the Technique of Semantic Segmentation

Jichiang Tsai <sup>1</sup>, Che-Cheng Chang <sup>2,\*</sup>  and Tzu Li <sup>3</sup>

<sup>1</sup> Department of Electrical Engineering & Graduate Institute of Communication Engineering, National Chung Hsing University, Taichung 402, Taiwan

<sup>2</sup> Department of Information Engineering and Computer Science, Feng Chia University, Taichung 407, Taiwan

<sup>3</sup> Department of Electrical Engineering, National Chung Hsing University, Taichung 402, Taiwan

\* Correspondence: chechang@fcu.edu.tw; Tel.: +886-4-24517250 (ext. 3764)

**Abstract:** Advanced Driver Assistance Systems (ADAS) are only applied to relatively simple scenarios, such as highways. If there is an emergency while driving, the driver should take control of the car to deal properly with the situation at any time. Obviously, this incurs the uncertainty of safety. Recently, in the literature, several studies have been proposed for the above-mentioned issue via Artificial Intelligence (AI). The achievement is exactly the aim that we look forward to, i.e., the autonomous vehicle. In this paper, we realize the autonomous driving control via Deep Reinforcement Learning (DRL) based on the CARLA (Car Learning to Act) simulator. Specifically, we use the ordinary Red-Green-Blue (RGB) camera and semantic segmentation camera to observe the view in front of the vehicle while driving. Then, the captured information is utilized as the input for different DRL models so as to evaluate the performance, where the DRL models include DDPG (Deep Deterministic Policy Gradient) and RDPG (Recurrent Deterministic Policy Gradient). Moreover, we also design an appropriate reward mechanism for these DRL models to realize efficient autonomous driving control. According to the results, only the RDPG strategies can finish the driving mission with the scenario that does not appear/include in the training scenario, and with the help of the semantic segmentation camera, the RDPG control strategy can further improve its efficiency.

**Keywords:** autonomous driving; deep deterministic policy gradient; recurrent deterministic policy gradient; semantic segmentation



**Citation:** Tsai, J.; Chang, C.-C.; Li, T. Autonomous Driving Control Based on the Technique of Semantic Segmentation. *Sensors* **2023**, *23*, 895. <https://doi.org/10.3390/s23020895>

Academic Editors: Angel Valera and Francisco Valero

Received: 29 October 2022

Revised: 31 December 2022

Accepted: 10 January 2023

Published: 12 January 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The urban population is growing, and this increase has different outgrowths concerning different topics. For example, the present and further increase in traffic flow is affecting the operational and safety performance of roadways. In accordance with international traffic statistics, the number of daily traveling vehicles amounts to one billion and will reach four billion in 2050 [29]. Hence, automotive safety has become a very critical issue. The automotive safety improvements in the past are all passive safety designs to minimize injuries during accidents, e.g., airbags, shatter-resistant glass, and so on. Then Advanced Driver Assistance Systems (ADAS) actively improve automotive safety by controlling both accelerating/decelerating and steering to reduce the occurrence of accidents and injuries. For instance, in [30,31], the authors propose a car-following model, which works in a human-like way from speed, relative speed, and inter-vehicle spacing. However, ADAS can only be applied to relatively simple scenarios such as highways. If there is an emergency while driving, the driver should take control of the car to deal properly with the situation at any time. Obviously, this incurs the uncertainty of safety. Notice that the main difference between passive and active safety is the operating time [32]. That is to say, an active safety system will operate before the accident and thus attempt to avoid such an accident.

Recently, in the literature, there have been several studies proposed for the above-mentioned issue via Artificial Intelligence (AI). They utilize dynamic simulators with

the capacity to accurately and efficiently simulate the behavior of vehicles to conduct experiments and algorithms. The achievement is exactly the aim that we look forward to, i.e., the autonomous vehicle [33–36]. More specifically, in [33], the authors utilize the transformation in diverse color spaces to design their reward mechanism of Deep Reinforcement Learning (DRL) algorithms. Since the Hue-Saturation-Value (HSV) model is more closely aligned with the color-making attributes of human vision than the Red-Green-Blue (RGB) model, the former is better for color gradations found in nature [37,38]. In [35], the lidar and odometer data are used to obtain the distance and heading information for the designs of their reward mechanism of DRL algorithms. Consequently, we can realize that in the literature, all existing DRL algorithms are only frameworks; it is needed to design and experiment elaborately to implement a viable model based on a certain concept for a specific application.

Likewise, in this paper, we realize autonomous driving control via DRL as well. The CARLA (Car Learning to Act) simulator [39] provides all the necessary components for our experiments, e.g., the car module, the road scenes, the sensed information, and so forth. Specifically, we use an ordinary RGB camera and a semantic segmentation camera to observe the view in front of the vehicle while driving. Notice that since CARLA provides the component of a semantic segmentation camera for the simulation, we do not need to contemplate its implementation; it can be treated as a kind of independent sensor. Then the captured information is utilized as the input for different DRL models so as to evaluate the performance, where the DRL models include DDPG (Deep Deterministic Policy Gradient) and RDPG (Recurrent Deterministic Policy Gradient). Moreover, we also design an appropriate reward mechanism for these DRL models to realize efficient autonomous driving control.

The rest of this paper is structured as follows. At the beginning, the introductory knowledge with regard to this work is reviewed in Section 2, e.g., CARLA, DRL, DDPG, RDPG, and so on. After that, our designs are detailed in Section 3, and the experiments and results corresponding to various experimental settings are performed, analyzed, and discussed in Section 4. Specifically, by utilizing the real-time images of the road in front of the vehicle captured by the ordinary RGB camera and semantic segmentation camera provided by CARLA as the training data, we formulate a proper reward generation architecture with the aim to improve the performance of the adopted models for the autonomous vehicle. Lastly, in Section 5, this work is concluded, and some possible futures are proposed.

## 2. Preliminary

### 2.1. CARLA Simulator

CARLA is an autonomous driving simulator that is open-source and provides the flexible Application Programming Interface (API) to address a wide range of driving tasks. Thus it can help democratize the development of autonomous driving, e.g., driving policies, perception algorithms, and so forth [39].

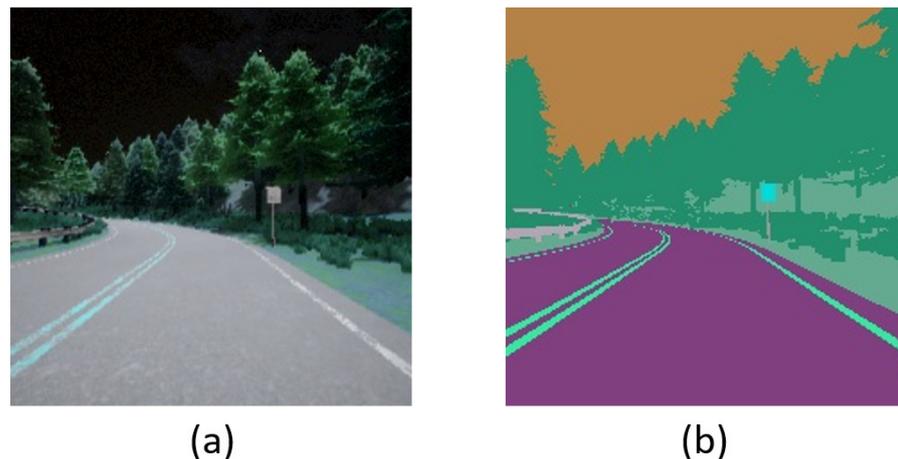
On the other hand, CARLA is based on Unreal Engine [40] and OpenDRIVE [41] to realize the simulation and define the settings of roads and areas. Noteworthy, Unreal Engine is a research and development tool for real-time technology, which gives creators the freedom and control of virtual worlds. Similarly, OpenDRIVE is used to furnish the road network description, which can be fed into the simulation to develop and validate new techniques.

### 2.2. RGB Camera and Semantic Segmentation Camera

An RGB camera is a camera equipped with a standard sensor through which the color images are acquired. It captures light in red, green, and blue wavelengths for color representation. Since RGB channels roughly follow human vision, it is generally utilized in computer displays.

On the other hand, a semantic segmentation camera is very promising and possesses several advantages over conventional cameras [42]. Each object is classified and colored

according to its class. For instance, buildings appear in a different color than pedestrians [39]. Moreover, it is a natural motion detector and can filter out redundant information automatically [43]. It can be implemented via various techniques, e.g., Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Generative Adversarial Network (GAN), and so on. Note that since CARLA provides the component of a semantic segmentation camera for the simulation, we do not need to contemplate its implementation; it can be treated as a kind of independent sensor. Figure 1 shows an example of driving vision obtained by the RGB camera and the semantic segmentation camera.



**Figure 1.** An example of driving vision: (a) RGB camera; (b) Semantic segmentation camera.

### 2.3. Constituents of Reinforcement Learning

The main concept of Reinforcement Learning (RL) is the interaction between the agent and the environment. In the process of two-way effect, four primary components need to be considered, i.e., the policy, the reward, the value function, and the environmental model [44–46].

- The policy: It is a mapping of all perceived environmental states to actions that can be taken during the procedure. It may involve some extended computations or may be a simple lookup table.
- The reward: After each step, the agent will be rewarded with a single number, which is called the reward. Maximizing the total reward during the procedure is the ultimate object. Noteworthy, a step may simultaneously include more than one action for some particular application.
- The value function: The reward only reveals what decision is better or worse instantly. However, in the long term, the value function can reveal what sequence of decisions is worthier or unworthier. That is to say, for a state, the value is the sum of the reward that an agent may amass from now on.
- The environmental model: The environmental model is utilized to mimic the environmental behavior. Via the environmental model, we can infer the possible future before the experiment is actually performed. Namely, it is utilized for preplanning. Noteworthy, a model-based method needs an environmental model to infer the possible future, while a model-free method is simply trial-and-error.

### 2.4. Model-Based and Model-Free Methods

The model-based method has extreme effectiveness in some applications where the environments are known previously from their rules or designs. Nevertheless, fully model-based cases are rare or may only fit simple and useless scenarios. Namely, the pure tabular solution will limit applicability. On the other hand, the model-free method makes decisions by its knowledge that has been learned via the strategy of trial and error. Apparently, its performance was poor initially due to limited knowledge. However, with continued

experiences, it will become more trustworthy. Hence, we can conclude that the model-based method depends on planning while the model-free method depends on learning.

According to the above discussions, we have comprehended that since the environmental model for a model-based solution should be accurate enough to be useful, a model-free solution will have more advantages in the practical and complex applications. That is to say, we will not have the problem of creating a sufficiently accurate environmental model because the model-free solution is used. Noteworthy, the model-free strategy depends on stored values with regard to state-action pairs. They are estimations that the agent can anticipate starting from every action taken at every state. These are acquired via a lot of trials from start to finish. Finally, when the process becomes good enough, the agent will only select the action with the largest action value at every state to make optimal decisions [44].

There are several kinds of model-free branches proposed in the literature, e.g., value-based DQN, policy-based DDPG, policy-based RDPG, and so on. More specifically, since DQN estimates all values for every state-action pair, its performance falls while the considered action space is continuous/complex. Nevertheless, the issue is dealt with easily via DDPG, which possesses the nature of meeting the high dimensional data inherited from Deterministic Policy Gradient (DPG) [47,48]. RDPG is also an extended version of DPG. For solving various physical control problems properly, the memory concept is introduced into DPG. Namely, the short-term integration of information and the long-term integration of memory are both considered in the RDPG approach [49]. In this paper, we will use DDPG and RDPG, combined with different camera data, to implement the autonomous control strategies in Section 3.

### 2.5. Experience Replay

Q-learning, including its successors, can be improved via a method called experience replay [50,51]. It stores the experience at every step in a replay memory. After the emulator executes an action  $a_t$  in a state  $s_t$ , the reward  $r_{t+1}$  and state  $s_{t+1}$  will be returned. They form the tuple  $(s_t, a_t, r_{t+1}, s_{t+1})$  and are stored in the replay memory. Here, we can accumulate experiences via a lot of plays of the same experiment or game. Then, a mini-batch sampled uniformly and randomly from the replay memory is performed for updates. That is to say, instead of  $s_{t+1}$  becoming the update of  $s_t$ , a new unconnected experience is drawn to supply data for the next update.

With the concept of experience replay, Q-learning provides several advantages over its usual form. It allows Q-learning, including its successors, to learn more efficiently. Namely, experience replay reduces the variance of updates and eliminates one source of instability.

### 2.6. Target Network

The concept of the target network is with two opposite networks. One is a target network, which should be relatively stable. The other is being updated continuously. The target network copies the weight from the latter after some predefined steps to avoid the irrelevance of target estimates. Obviously, the former is like offline, and the latter is like online [52].

On the other hand, it can be implemented in another way, i.e., a shared deep learning architecture. Particularly, in the neural network with shared architecture, some common layers are placed on the upper part, and then some split layers are accordingly cascaded onto the lower part for different dedicated purposes. Namely, one is for computing the state value, and the other is for computing the advantage. After that, both are combined to get the action-value approximation.

### 2.7. Policy Gradient

The policy gradient-based approach is very popular, which attempts to leverage the property of differentiability of the policy function for optimization. Particularly, since the policy term is parameterized, it is stochastic. Namely, for a given state, instead of the single

best action, the policy may have various probabilities of choosing distinct actions. Hence, it draws stochastic samples to refine the estimate so as to optimize the policy, which may enable the agent to accumulate the maximum cumulative reward [52].

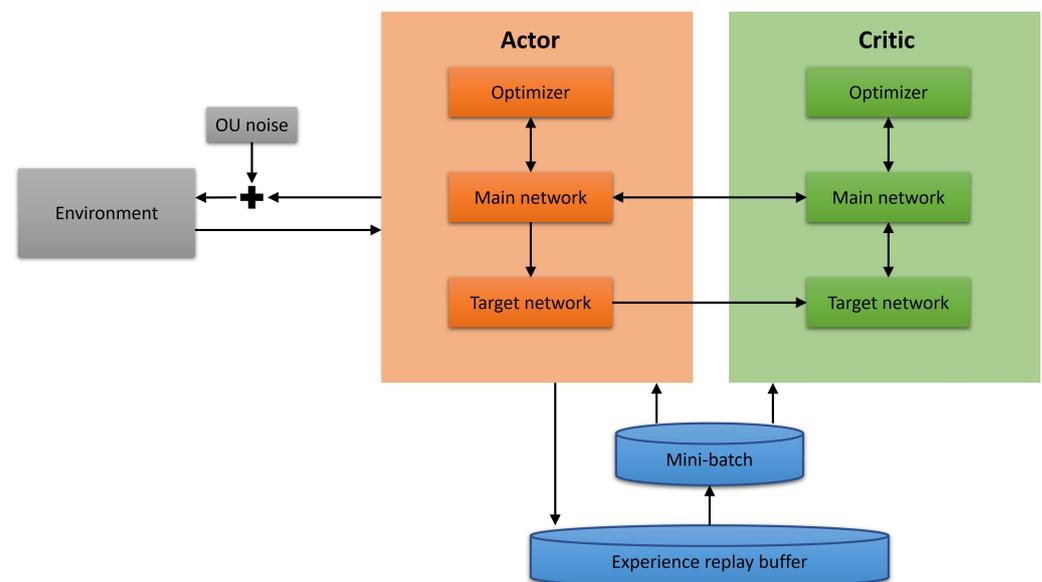
### 2.8. OU Noise

The OU (Ornstein–Uhlenbeck) process is a stochastic process with applications in physical sciences. Originally, it was a model for the velocity of a massive particle under the influence of friction. Noteworthy, the OU process has the property of stationary Gauss–Markov; it may be a Gaussian process, a Markov process, or a homogeneous one [53].

Since OU noise is with the property of time sequence, for the application of vehicle inertia, which is the resistance to change the velocity or direction of a vehicle, either in motion or at rest [54], OU noise is better than Gaussian noise.

### 2.9. DRL Architecture

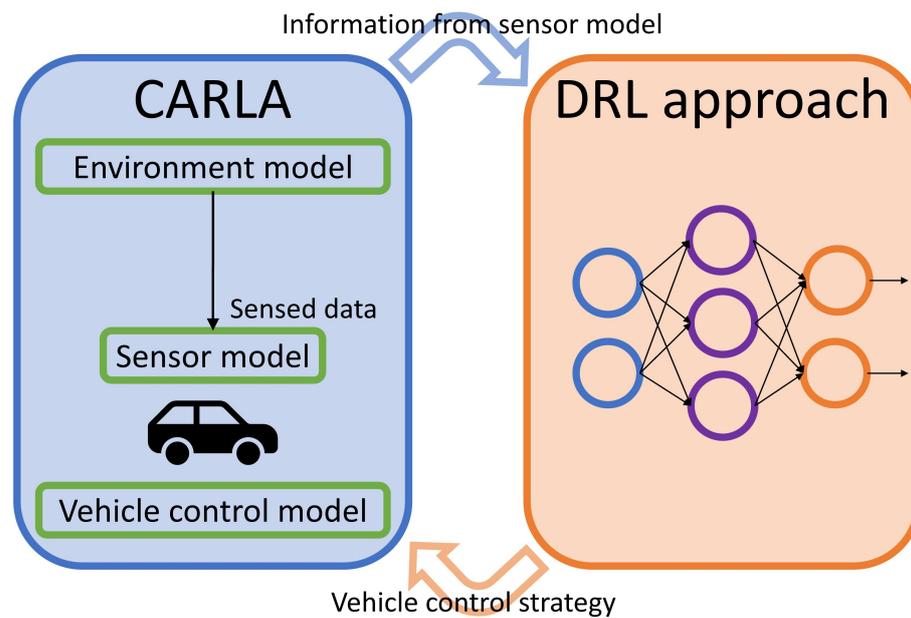
According to the aforementioned descriptions, we can conduct the architecture shown in Figure 2, which includes the experience replay buffer, the introduction of the target network and OU noise, and so on. Owing to the design of the architecture, for various DRL approaches, we only need to replace the source code in the network part with another one.



**Figure 2.** The DRL architecture.

## 3. The DRL Control Strategies

Our autonomous driving control strategies are elaborated on in this section. Further, as the description in [33,35], all existing DRL notions proposed in the literature are just frameworks, thus designing a reward mechanism and experimenting with such a mechanism to realize a particular application is still needed. This is the primary contribution of our work. Here, the diagram illustrated in Figure 3 is to demonstrate the relationship between the CARLA simulator and our autonomous driving control strategies. Owing to the design of simulation architecture, for various DRL approaches, we only need to replace the source code in the network part (right component) with another. The CARLA simulator (left component) produces training data that are sent to the DRL approach for the training process. After the training process is finished, the right component is changed to continuously make immediate responses to the left component for controlling the vehicle according to its real-time driving situation.



**Figure 3.** The relationship between CARLA and the control strategy.

### 3.1. Designs of Reward Mechanism

In this section, we start to introduce the six primary constituents of the reward mechanism, i.e.,  $R_{area}$ ,  $R_{roadline}$ ,  $R_{waypoint}$ ,  $R_{velocity}$ ,  $R_{steer}$ , and  $R_{punishment}$ . Noteworthy, the design of the reward mechanism plays an important role in the DRL approach, which affects the performance of convergence as well as the accuracy and stability of the interaction between the agent and the environment.

First, if a vehicle follows the road stably and accurately, the driving vision will include a higher percentage of the road area. Oppositely, if the driving vision includes a lower percentage of the road area, this means that the vehicle can not follow the road properly. Namely, it may currently be out of control. Hence, according to the notion, the first constituent of the reward mechanism is as follows:

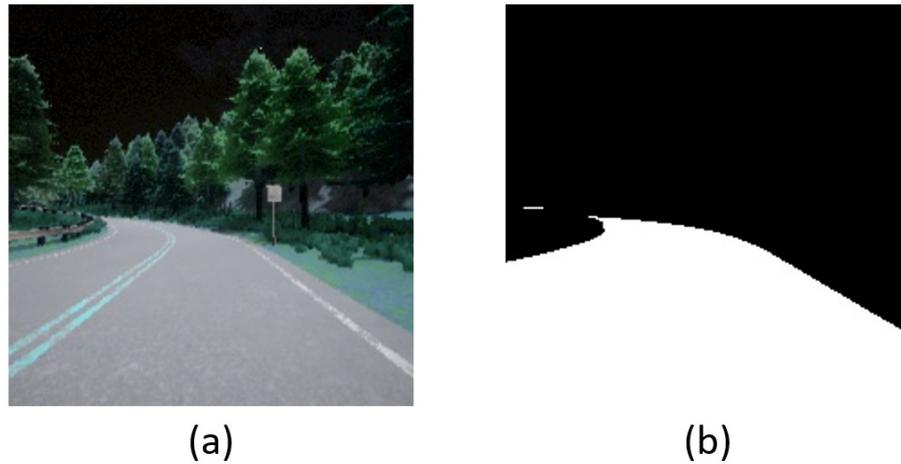
$$R_{area} = (A_{road} - T_{road}) \times w_{road}, \quad (1)$$

where  $A_{road}$  is the measure of road area,  $T_{road}$  is the threshold, and  $w_{road}$  is the weight. Here,  $w_{road}$  is used for regulating the ratio of reward. Traditionally, we can use computer vision skill to obtain  $A_{road}$ . Particularly, since the HSV color model has better efficacy for color gradations found in nature, the first step is to convert the RGB data into HSV color space. After that, binarization replaces each pixel in the driving vision with a white/black pixel [55,56] to obtain  $A_{road}$ . Noteworthy, thanks to the characteristics of the semantic segmentation camera, in this paper, we can receive  $A_{road}$  without the transformation of color space. Figure 4 shows an example of using driving vision to obtain  $A_{road}$ .

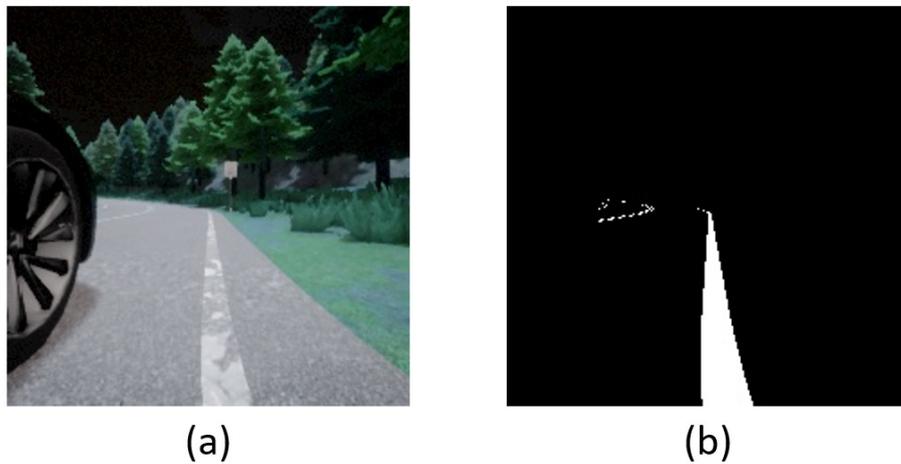
Second, the constituent is  $R_{roadline}$ , which is used to ensure the keeping of the vehicle inside the lane. If the vehicle stays in its lane, the higher  $R_{roadline}$  is rewarded, and vice versa.

$$R_{roadline} = \begin{cases} C_{line} \times w_{line}, & (T_{line,x_1} < I_{line,x} < T_{line,x_2}) \& (T_{line,y} < I_{line,y}) \\ -C_{line} \times w_{line}, & otherwise \end{cases}, \quad (2)$$

where  $C_{line}$  is a constant,  $T_{line,x_1}$ ,  $T_{line,x_2}$ , and  $T_{line,y}$  are the thresholds of the driving vision,  $I_{line,x}$  and  $I_{line,y}$  are the current coordinates of the lane lines in the side driving vision, and  $w_{line}$  is the weight. Figure 5 shows an example of using side driving vision to obtain lane lines. Notice that the parameters of binarization are different from Figure 4.



**Figure 4.** An example of driving vision to obtain  $A_{road}$ : (a) Original; (b) Binarization.



**Figure 5.** An example of side driving vision to obtain lane lines: (a) Original; (b) Binarization.

The next constituent is  $R_{waypoint}$ , which is used to ensure correct lane keeping. Because the scenario is right-hand traffic (RHT), the vehicle should keep to the right-hand side lane. Here, the concept of navigation with waypoints is adopted to avoid driving on the wrong side of the road. Particularly, before each trip, the sequence of waypoints will be generated via navigation in advance. Therefore, the vehicle will chase these waypoints for correct lane keeping as well as a higher reward.

$$R_{waypoint} = (-|P_{vehicle} - P_{waypoint}|^2 + T_{waypoint}) \times w_{waypoint}, \quad (3)$$

where  $P_{vehicle}$  is the position of the vehicle,  $P_{waypoint}$  is the position of the next waypoint,  $T_{waypoint}$  is the threshold, and  $w_{waypoint}$  is the weight. Figure 6 shows an example of the sequence of waypoints generated via navigation.



**Figure 6.** An example of the sequence of waypoints generated via navigation.

The next two constituents are regarding the stability of the vehicle, i.e.,  $R_{velocity}$  and  $R_{steer}$ . Particularly, if the vehicle speed is slower than a predefined value and the acceleration is not enough, it will be rewarded a negative award. In addition, if the steering angle is similar to that at the latest point in time, it will be rewarded a higher award. Therefore, we can obtain the following two designs for reward constituents.

$$R_{velocity} = \begin{cases} -C_{vel} \times w_{vel}, (V_{vel} < T_{vel}) \& (V_{thr} < T_{thr}) \\ 0, otherwise \end{cases}, \quad (4)$$

$$R_{steer} = (|S_t - S_{t-1}| - T_{steer}) \times w_{steer}, \quad (5)$$

where  $C_{vel}$  is a constant,  $w_{vel}$  is the weight,  $V_{vel}$  is the current velocity of the vehicle,  $T_{vel}$  is the predefined threshold of velocity,  $V_{thr}$  is the current throttle value of the vehicle,  $T_{thr}$  is the predefined threshold of throttle value,  $S_t$  and  $S_{t-1}$  are the steering angles at times  $t$  and  $t - 1$ , and  $w_{steer}$  is the weight.

The last constituent is the punishment,  $R_{punishment}$ . Specifically, violating a traffic regulation causes a negative reward and then resets the experiment to the initial status. The constituent is presented below:

$$R_{punishment} = \begin{cases} -C_{punishment}, experiment\ reset \\ 0, otherwise \end{cases}, \quad (6)$$

where  $C_{punishment}$  is a constant.

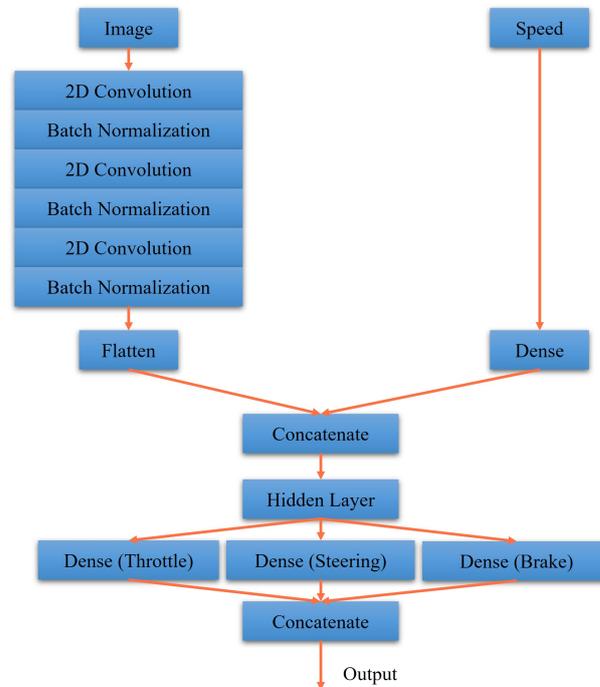
Eventually, the whole reward mechanism is constructed:

$$R_{total} = R_{area} + R_{roadline} + R_{waypoint} + R_{velocity} + R_{steer} + R_{punishment}. \quad (7)$$

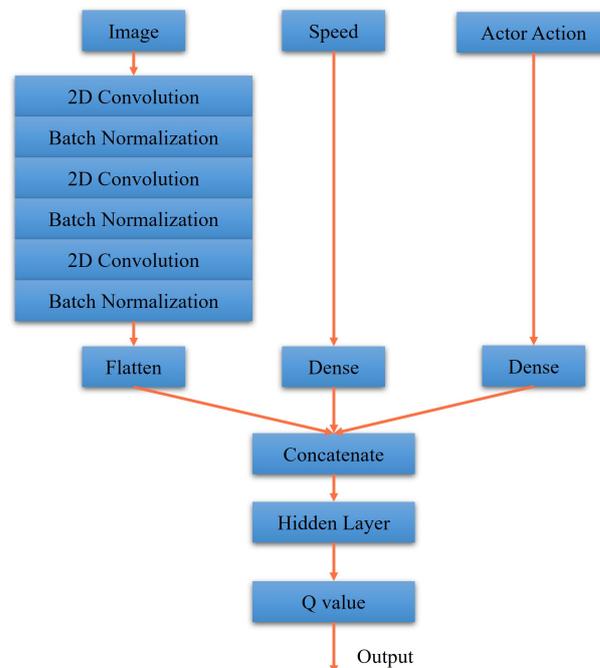
### 3.2. Designs of Actor-Critic Network

The actor and critic networks of DDPG and RDPG are shown in Figures 7–10. First, in Figure 7, the driving vision is taken as the input for the actor network of DDPG. After executing the process of 2D convolution and batch normalization three times, it is concatenated to the driving speed. After that, the concatenated data, including throttle, steering, and brake data, are reweighed as the output. In Figure 8, the critic network of DDPG is presented, where the driving vision, driving speed, and actor action is taken as the inputs. Afterward, they are concatenated and reweighed to obtain the Q value.

Similarly, Figure 9 shows the actor network of RDPG. Particularly, the historical data are taken as the inputs, i.e., driving vision and driving speed. After that, the Long Short-Term Memory (LSTM) layer is utilized to introduce the memory property. In Figure 10, also with the LSTM layer, the historical driving vision, driving speed, and actor action are taken as the inputs. Afterward, they are concatenated and reweighed to get the Q value.



**Figure 7.** Our actor network for DDPG.



**Figure 8.** Our critic network for DDPG.

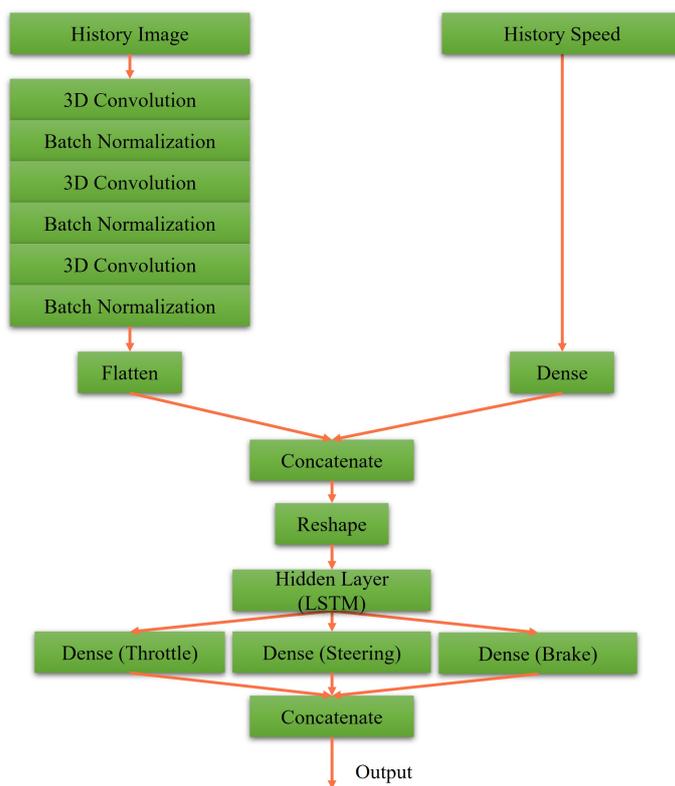


Figure 9. Our actor network for RDPG.

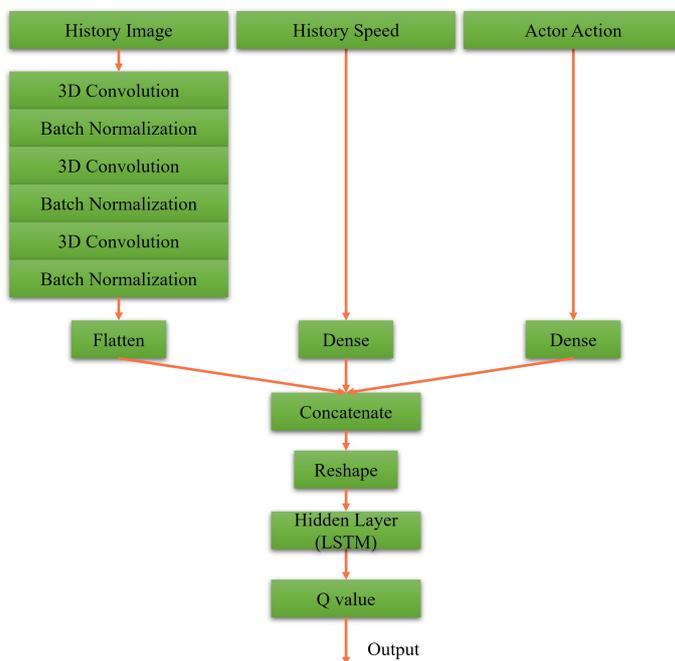


Figure 10. Our critic network for RDPG.

#### 4. Experimental Results

In this section, we start to address our experiments in detail. First, the path shown in Figure 11 is used for the training procedure, where the orange point is the starting point and the yellow the destination. In the same map, two other paths are used for the testing procedures. Note that parts of the training and testing paths are mutually exclusive; this is significant and critical for a fair assessment of the experiment. The simulation settings are shown below.

- Two DRL algorithms are adopted for the experiments, i.e., DDPG and RDPG.
- Two kinds of cameras are used to capture the driving vision for the experiments, i.e., RGB and semantic segmentation cameras.
- Hyperparameters:
  - Replay buffer of DDPG: 15,000.
  - Threshold of replay buffer of DDPG: 500.
  - Batch size of DDPG: 150.
  - Replay buffer of RDPG: 6000.
  - Threshold of replay buffer of RDPG: 500.
  - Batch size of RDPG: 150.
  - Learning rate: 0.0001 (actor) and 0.001 (critic).
  - Learning rate decay: 0.9.
  - $\epsilon$  from the start: 1.
  - $\epsilon$  decay: 0.99.
  - Minimum  $\epsilon$ : 0.01.
  - $\tau$ : 0.005.
  - $\mu$  of OU noise: 0.2 (throttle) and 0 (other).
  - $\theta$  of OU noise: 0.35.
  - $\sigma$  of OU noise: 0.1 (throttle) and 0.2 (other).
- The specification of the computer:
  - CPU: Intel Core i7-11700KF.
  - GPU: Nvidia GeForce RTX 3090 24GB.
  - RAM: 48GB DDR4 3600MHz.
  - HDD: 512GB SSD.
  - OS: Windows 10.

Notice that since RDPG has a memory property, less RDPG data occupies the same space as a larger number of DDPG data.



Figure 11. The map used for the training procedures.

Foremost, the first testing path is simpler and shorter, and the corresponding results are presented in Figures 12–15, where the orange point is the starting point and the yellow the destination. We can observe that the four autonomous vehicles based on different strategies all finish the driving mission. After that, the second path is more complex and longer, and the corresponding results are presented in Figures 16–19, where the orange point is the

starting point and the yellow the destination. Here, only two autonomous vehicles based on the RDPG strategies finish the driving mission. Two autonomous vehicles based on the DDPG strategies do not pass the scenario of an L-turn (both fail at the black point), which does not appear/include in the training scenario. According to the aforementioned results, we can conclude that since RDPG has the memory property of adopting inputs in sequence, it has better adaptability for a scenario that the autonomous vehicle has never seen in advance.



**Figure 12.** DDPG with an RGB camera for the first testing path.



**Figure 13.** DDPG with a semantic segmentation camera for the first testing path.



**Figure 14.** RDPG with an RGB camera for the first testing path.



Figure 15. RDPG with a semantic segmentation camera for the first testing path.



Figure 16. DDPG with an RGB camera for the second testing path.



Figure 17. DDPG with a semantic segmentation camera for the second testing path.

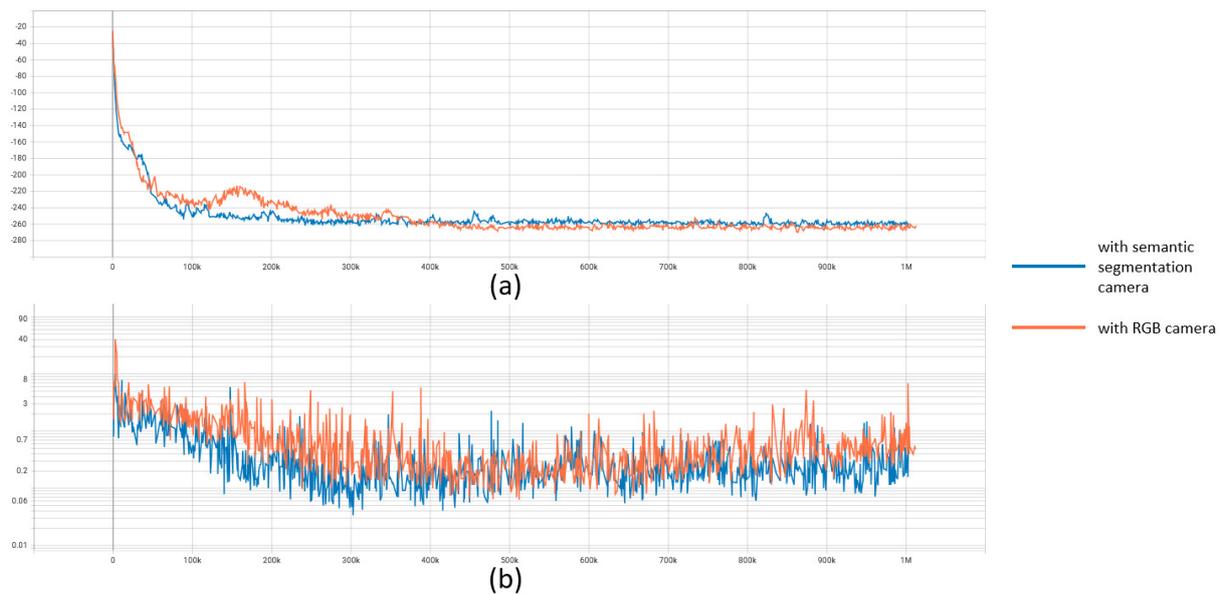


Figure 18. RDPG with an RGB camera for the second testing path.



Figure 19. RDPG with a semantic segmentation camera for the second testing path.

Next, we further discuss the training performance of two RDPG strategies that have better adaptability in the above experiments. The results are presented in Figure 20, where the actor loss and critic loss of RDPG during the training procedure are given. According to the data of the actor loss and critic loss, we can observe that RDPG with a semantic segmentation camera has better convergency performance. Even RDPG with a semantic segmentation camera has a lower loss value at the end of the training procedure. Hence, we can conclude that with the help of a semantic segmentation camera, RDPG can further improve its efficiency.



**Figure 20.** (a) The actor loss of RDPG during the training procedure; (b) The critic loss of RDPG during the training procedure.

## 5. Conclusions and Future Work

In this paper, we have used an ordinary RGB camera and a semantic segmentation camera to observe the view in front of the vehicle while driving. The captured information has also been utilized as the input for different DRL models so as to evaluate the performance, where the DRL models include DDPG and RDPG. Moreover, we have designed an appropriate reward mechanism for these DRL models to realize efficient autonomous driving control. According to the results, only the RDPG strategies can finish the driving mission with the scenario that does not appear/include in the training scenario. Next, we have further compared the RDPG strategies with different cameras, and the results have shown that with the help of a semantic segmentation camera, RDPG can further improve its efficiency.

For future work, we plan to propose some new methods with asymmetric architecture for achieving better performance. Namely, some data may be checked continuously, but some may be periodic. According to this asymmetric architecture, we will have less computational complexity and better convergence performance. On the other hand, for different types of vehicles, the consideration of cameras with different angles will be another critical topic for DRL control strategies.

**Author Contributions:** Conceptualization, J.T. and T.L.; methodology, J.T. and T.L.; software, T.L.; validation, J.T., C.-C.C. and T.L.; formal analysis, J.T., C.-C.C. and T.L.; writing—original draft preparation, J.T. and C.-C.C.; writing—review and editing, J.T. and C.-C.C.. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by National Science and Technology Council, Taiwan, R.O.C. under grant 109-2221-E-035-067-MY3.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

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

## References

1. Cafiso, S.; Graziano, A.D.; Giuffrè, T.; Pappalardo, G.; Severino, A. Managed Lane as Strategy for Traffic Flow and Safety: A Case Study of Catania Ring Road. *Sustainability* **2022**, *14*, 2915.
2. Zhu, M.; Wang, X.; Wang, Y. Human-Like Autonomous Car-Following Model with Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1901.00569.
3. Zhu, M.; Wang, Y.; Pu, Z.; Hu, J.; Ke, X.W.R. Safe, Efficient, and Comfortable Velocity Control Based on Reinforcement Learning for Autonomous Driving. *Transp. Res. Part Emerg. Technol.* **2020**, *117*, 102662.
4. Chang, C.-C.; Chan, K.-L. Collision Avoidance Architecture Based on Computer Vision with Predictive Ability. In Proceedings of the 2019 International Workshop of ICAROB—Intelligent Artificial Life and Robotics, Beppu, Japan, 10–13 January 2019. [\[CrossRef\]](#)
5. Chang, C.-C.; Tsai, J.; Lin, J.-H.; Ooi, Y.-M. Autonomous Driving Control Using the DDPG and RDPG Algorithms. *Appl. Sci.* **2021**, *11*, 10659.
6. Home-AirSim [Online]. Available online: <https://microsoft.github.io/AirSim/> (accessed on 20 October 2022). [\[CrossRef\]](#)
7. Tsai, J.; Chang, C.-C.; Ou, Y.-C.; Sieh, B.-H.; Ooi, Y.-M. Autonomous Driving Control Based on the Perception of a Lidar Sensor and Odometer. *Appl. Sci.* **2022**, *12*, 7775.
8. Gazebo [Online]. Available online: <http://gazebosim.org/> (accessed on 20 October 2022). [\[CrossRef\]](#)
9. Agoston, M.K. *Computer Graphics and Geometric Modeling: Implementation and Algorithms*; Springer: London, UK, 2005.
10. Cheng, H.D.; Jiang, X.H.; Sun, Y.; Wang, J. Color Image Segmentation: Advances and Prospects. *Pattern Recognit.* **2001**, *34*, 2259–2281. [\[CrossRef\]](#)
11. CARLA Simulator [Online]. Available online: <https://carla.org/> (accessed on 20 October 2022).
12. The Most Powerful Real-Time 3D Creation Platform—Unreal Engine [Online]. Available online: <https://www.unrealengine.com/en-US/> (accessed on 20 October 2022).
13. ASAM OpenDRIVE [Online]. Available online: <https://www.asam.net/standards/detail/opendrive/> (accessed on 20 October 2022). [\[CrossRef\]](#)
14. Alonso, I.; Murillo, A.C. EV-SegNet: Semantic Segmentation for Event-Based Cameras. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 16–17 June 2019.
15. Maqueda, A.I.; Loquercio, A.; Gallego, G.; Garcia, N.; Scaramuzza, D. Event-Based Vision Meets Deep Learning on Steering Prediction for Self-Driving Cars. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018.
16. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; The MIT Press: Cambridge, UK, 2018.
17. Martin-Guerrero, J.D.; Lamata, L. Reinforcement Learning and Physics. *Appl. Sci.* **2021**, *11*, 8589.
18. Jembre, Y.Z.; Nugroho, Y.W.; Khan, M.T.R.; Attique, M.; Paul, R.; Shah, S.H.A.; Kim, B. Evaluation of Reinforcement and Deep Learning Algorithms in Controlling Unmanned Aerial Vehicles. *Appl. Sci.* **2021**, *11*, 7240.
19. Deep Reinforcement Learning [Online]. Available online: <https://julien-vitay.net/deeprl/> (accessed on 20 October 2022).
20. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous Control with Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1509.02971. [\[CrossRef\]](#)
21. Heess, N.; Hunt, J.J.; Lillicrap, T.P.; Silver, D. Memory-based Control with Recurrent Neural Networks. *arXiv* **2015**, arXiv:1512.04455. [\[CrossRef\]](#)
22. 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*, 8503.
23. Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.* **1992**, *8*, 293–321.
24. Sewak, M. *Deep Reinforcement Learning*; Springer: Cham, Switzerland, 2019.
25. Bibbona, E.; Panfilo, G.; Tavella, P. The Ornstein-Uhlenbeck process as a model of a low pass filtered white noise. *Metrologia* **2008**, *45*, S117. [\[CrossRef\]](#)
26. Vehicle Dynamics [Online]. Available online: <https://ritzel.siu.edu/courses/302s/vehicle/vehicledynamics.htm> (accessed on 20 October 2022). [\[CrossRef\]](#)
27. Chaki, N.; Shaikh, S.H.; Saeed, K. *Exploring Image Binarization Techniques*; Springer: Berlin/Heidelberg, Germany, 2014.
28. Stockman, G.; Shapiro, L.G. *Computer Vision*; Prentice Hall PTR: Hoboken, NJ, USA, 2001. [\[CrossRef\]](#)
29. Cafiso, S.; Graziano, A.D.; Giuffrè, T.; Pappalardo, G.; Severino, A. Managed Lane as Strategy for Traffic Flow and Safety: A Case Study of Catania Ring Road. *Sustainability* **2022**, *14*, 2915.
30. Zhu, M.; Wang, X.; Wang, Y. Human-Like Autonomous Car-Following Model with Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1901.00569.
31. Zhu, M.; Wang, Y.; Pu, Z.; Hu, J.; Ke, X.W.R. Safe, Efficient, and Comfortable Velocity Control Based on Reinforcement Learning for Autonomous Driving. *Transp. Res. Part Emerg. Technol.* **2020**, *117*, 102662.
32. Chang, C.-C.; Chan, K.-L. Collision Avoidance Architecture Based on Computer Vision with Predictive Ability. In Proceedings of the 2019 International Workshop of ICAROB—Intelligent Artificial Life and Robotics, Beppu, Japan, 10–13 January 2019. [\[CrossRef\]](#)
33. Chang, C.-C.; Tsai, J.; Lin, J.-H.; Ooi, Y.-M. Autonomous Driving Control Using the DDPG and RDPG Algorithms. *Appl. Sci.* **2021**, *11*, 10659.

34. Home-AirSim [Online]. Available online: <https://microsoft.github.io/AirSim/> (accessed on 20 October 2022). [CrossRef]
35. Tsai, J.; Chang, C.-C.; Ou, Y.-C.; Sieh, B.-H.; Ooi, Y.-M. Autonomous Driving Control Based on the Perception of a Lidar Sensor and Odometer. *Appl. Sci.* **2022**, *12*, 7775.
36. Gazebo [Online]. Available online: <http://gazebo.org/> (accessed on 20 October 2022). [CrossRef]
37. Agoston, M.K. *Computer Graphics and Geometric Modeling: Implementation and Algorithms*; Springer: London, UK, 2005.
38. Cheng, H.D.; Jiang, X.H.; Sun, Y.; Wang, J. Color Image Segmentation: Advances and Prospects. *Pattern Recognit.* **2001**, *34*, 2259–2281. [CrossRef]
39. CARLA Simulator [Online]. Available online: <https://carla.org/> (accessed on 20 October 2022).
40. The Most Powerful Real-Time 3D Creation Platform—Unreal Engine [Online]. Available online: <https://www.unrealengine.com/en-US/> (accessed on 20 October 2022).
41. ASAM OpenDRIVE [Online]. Available online: <https://www.asam.net/standards/detail/opendrive/> (accessed on 20 October 2022). [CrossRef]
42. Alonso, I.; Murillo, A.C. EV-SegNet: Semantic Segmentation for Event-Based Cameras. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 16–17 June 2019.
43. Maqueda, A.I.; Loquercio, A.; Gallego, G.; Garcia, N.; Scaramuzza, D. Event-Based Vision Meets Deep Learning on Steering Prediction for Self-Driving Cars. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018.
44. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; The MIT Press, Cambridge, UK, 2018.
45. Martin-Guerrero, J.D.; Lamata, L. Reinforcement Learning and Physics. *Appl. Sci.* **2021**, *11*, 8589.
46. Jembre, Y.Z.; Nugroho, Y.W.; Khan, M.T.R.; Attique, M.; Paul, R.; Shah, S.H.A.; Kim, B. Evaluation of Reinforcement and Deep Learning Algorithms in Controlling Unmanned Aerial Vehicles. *Appl. Sci.* **2021**, *11*, 7240.
47. Deep Reinforcement Learning [Online]. Available online: <https://julien-vitay.net/deeprl/> (accessed on 20 October 2022).
48. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous Control with Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1509.02971. [CrossRef]
49. Heess, N.; Hunt, J.J.; Lillicrap, T.P.; Silver, D. Memory-based Control with Recurrent Neural Networks. *arXiv* **2015**, arXiv:1512.04455. [CrossRef]
50. 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*, 8503.
51. Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.* **1992**, *8*, 293–321.
52. Sewak, M. *Deep Reinforcement Learning*; Springer: Cham, Switzerland, 2019.
53. Bibbona, E.; Panfilo, G.; Tavella, P. The Ornstein-Uhlenbeck process as a model of a low pass filtered white noise. *Metrologia* **2008**, *45*, S117. [CrossRef]
54. Vehicle Dynamics [Online]. Available online: <https://ritzelsiu.edu/courses/302s/vehicle/vehicledynamics.htm> (accessed on 20 October 2022). [CrossRef]
55. Chaki, N.; Shaikh, S.H.; Saeed, K. *Exploring Image Binarization Techniques*; Springer: Berlin/Heidelberg, Germany, 2014.
56. Stockman, G.; Shapiro, L.G. *Computer Vision*; Prentice Hall PTR: Hoboken, NJ, USA, 2001. [CrossRef]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.