Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 56-2 (2023) 7948–7953

Design of Reward Function on Reinforcement Learning for Automated Driving

Takeru Goto [*] **Yuki Kizumi** [**] **Shun Iwasaki** [**]

[*] *Innovative Research Excellence, Honda R&D Co., Ltd., Tokyo, Japan (e-mail: takeru_goto@jp.honda).*

[**] *Business Development Operations, Honda Motor Co., Ltd., Tokyo, Japan.*

Abstract: This paper proposes a design scheme of reward function that constantly evaluates both driving states and actions for applying reinforcement learning to automated driving. In the field of reinforcement learning, reward functions often evaluate whether the goal is achieved by assigning values such as +1 for success and -1 for failure. This type of reward function can potentially obtain a policy that achieves the goal, but the process by which the goal is reached is not evaluated. However, process to reach a destination is important for automated driving, such as keeping velocity, avoiding risk, retaining distance from other cars, keeping comfortable for passengers. Therefore, the reward function designed by the proposed scheme is suited for automated driving by evaluating driving process. The effects of the proposed scheme are demonstrated on simulated circuit driving and highway cruising. Asynchronous Advantage Actor-Critic is used, and models are trained under some situations for generalization. The result shows that appropriate driving positions are obtained, such as traveling on the inside of corners, and rapid deceleration to turn along sharp curves. In highway cruising, the ego vehicle becomes able to change lane in an environment where there are other vehicles with suitable deceleration to avoid catching up to a front vehicle, and acceleration so that a rear vehicle does not catch up to the ego vehicle.

Keywords: reinforcement learning, autonomous vehicle, reward function, asynchronous advantage Actor-Critic, parallel learning

1. INTRODUCTION

Reinforcement learning(RL) is a kind of machine learning that obtains an action policy which enables an agent to acquire greater rewards. Mnih et al. (2013, 2015) showed that deep reinforcement learning(DRL) which combines a deep neural network and reinforcement learning enables to obtain scores exceeding those of humans in some Atari games. This has promoted researches on DRL.

Meanwhile, appropriate reward functions are needed to generate appropriate policies. The game “Pong” is often used to evaluate RL algorithms, and the reward function commonly used in this case assigns a score of +1 when the opponent cannot return the ball, -1 when the actor cannot return the ball, and 0 for other states, e.g. Mnih et al. (2015). This paper refers to this type of reward function as a purpose-oriented reward function. Purpose-oriented reward functions are thought to be suited to tasks where the goal is clear and any ways of achieving that goal are acceptable. This means that goals may be achieved by actions that are not easily imaginable by humans.

However, in case of automated driving, it is not enough to simply arrive at the destination without accident. The vehicle needs to drive over the assigned route while maintaining an appropriate distance between vehicles, not coming too close to pedestrians, obeying traffic rules such as speed limits, and taking into account ride comfort. It is a challenge to obtain an appropriate policy for these types of tasks using +1/-1 type purpose-oriented rewards. This calls for a reward function that constantly evaluates the appropriateness of states and actions. This paper refers to this type of reward function as a process-oriented reward function. However, process-oriented reward functions have a high degree of freedom and it is a challenge to design a reward function that can obtain an appropriate policy.

There are a lot of works for applying RL to automated driving. As an early study, Stafylopatis and Blekas (1998) describes that the appropriate heading direction for avoiding walls can be learned with RL. Michels et al. (2005) realized obstacle avoidance. Recently, Schester and Ortiz (2021) proved that multi-agent DRL is effective for highway lamp merge. In these works, there are some types of rewards. The penalty on other objects or static obstacles is often used, e.g. Stafylopatis and Blekas (1998); Michels et al. (2005). Desjardins and Chaib-Draa (2011) adopts the distance from obstacles or center of lane as the similar reason. That can also consider the appropriate inter-vehicle distance and position in lane. In works where RL was applied to racing simulators, e.g. Lillicrap et al. (2015); Perot et al. (2017), reward functions were calculated from ego velocity and offset from the center of a road. As another point of view, drivers' comfortability is important for automated driving. Ye et al. (2020) considers the difference between actual and desired velocity, acceleration and jerk for that objective. These works use process-oriented reward functions partially, but each reward function is designed for each task.

Therefore, this paper proposes a general design scheme of process-oriented reward function for automated driving. This scheme calculates an evaluation value between 0 and 1 for all evaluation items, and determines the reward based on the product of those values. A separate evaluation function can be applied to each evaluation item, which facilitates a process-oriented reward function design. In addition, the proposed scheme is applied to circuit driving and highway cruising to investigate application to automated driving.

2. PREVIOUS RESEARCH

2.1 Asynchronous Advantage Actor-Critic

This paper uses Asynchronous Advantage Actor-Critic (A3C) developed by Mnih et al. (2016) as the RL algorithm. A3C is an Actor-Critic type algorithm, and can output continuous action. In addition, multiple agents simultaneously learn asynchronously, so this can utilize computation resources efficiently.

A3C has a policy function $\pi(a_t|s_t;\theta)$ and an estimated state value function $V(s_t;\theta_v)$. a_t and s_t are the respective action and state at time t, θ and θ_v are parameters that respectively determine the policy function and state value function. The action is a random variable that conforms to the policy function, and the probability of each action being selected is determined when the state is determined. The state value function is the expected value of the future return R in state s_t. Return R is the discounted sum of the rewards from time t onward, and is as follows.

$$R = \sum_{i=0}^{\infty} \gamma^i r_{t+i} \tag{1}$$

γ is the discount factor, and r_t is the reward given at time t. The policy function and state function are updated when an action has been performed t_{max} times or when the terminal state is reached. The policy function update amount is determined as $\nabla_{\theta'} \log \pi(a_t|s_t;\theta')A(s_t,a_t;\theta,\theta_v)$. The update amout of the state function is determined as $\partial A(s_t,a_t;\theta,\theta'_v)/\partial\theta'_v$. $A(s_t,a_t;\theta,\theta_v)$ is called the advantage function as follows.

$$A(s_t,a_t;\theta,\theta_v)=\sum_{i=0}^{k-1}\gamma^i r_{t+i}+\gamma^k V(s_{t+k};\theta_v)-V(s_t;\theta_v) \tag{2}$$

Learning is performed using multiple threads. The environment and agents operate asynchronously in each thread. The parameters θ and θ_v are stored in a shared memory. Each thread first copies the shared parameters to a local memory. Each agent uses the local parameters to determine the action, calculates the update amounts, updates the shared parameters, and then copies the updated shared parameters to the local memory again.

2.2 Example of Reward Functions

Two reward functions are introduced below as examples of applying RL to a racing simulator. The reward function in Lillicrap et al. (2015) is as follows.

$$r_t = \begin{cases} -1 & \text{(collision with wall)} \\ v_t \cos\theta_t & \text{(others)} \end{cases} \tag{3}$$

Here, v_t is the velocity in the vehicle traveling direction, and θ_t is the angle relative to the course traveling direction. The reward is process-oriented for velocity, but the penalty for collision is purpose-oriented. Therefore, learning to accelerate is easy, but learning to avoid collision is thought to take time, or easily become unstable. In fact, Perot et al. (2017) reports that despite driving a simple oval course, learning was unstable, with success in some cases and failure in others.

The reward function which is adopted by Perot et al. (2017) is as follows.

$$r_t = v_t(\cos\theta_t - d_t) \tag{4}$$

Here, d_t is the offset from the center of the course. This reward is completely process-oriented, and learning is thought to proceed quickly. In addition, it is a simple reward of more quickly traveling straight in the center of the course. However, it is difficult to adjust priorities, such as loosening the restriction to travel in the center and increasing the speed, as well as modifying the function, such as addition of a term related to ride comfort

3. DESIGN SCHEME OF PROCESS-ORIENTED REWARD FUNCTION

This chapter proposes a design scheme of a process-oriented reward function that is suited for automated driving. When there are multiple evaluation items, the priority can be adjusted and new evaluation items can also be added.

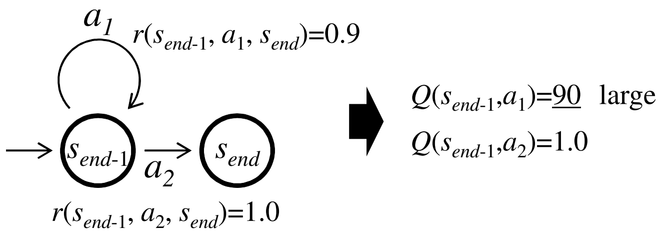
When designing the reward, first the evaluation items $f_k(s,a,s')$ and evaluation functions $e_k(f)$ are determined. Here, $k < n$, where n is the number of evaluation items. The evaluation items must be determined uniquely according to the states and actions. The evaluation function maps the evaluation items to evaluation values between 0 and 1. The temporary reward $r_{tmp}(s,a,s')$ is the product of all the evaluation values, and is expressed as follows.

$$r_{tmp}(s,a,s') = \prod_{k=0}^{n-1} e_k(f_k(s,a,s')) \tag{5}$$

The reward $r(s,a,s')$ is given as follows to help prevent state value from decreasing as the state approaches the terminal state.

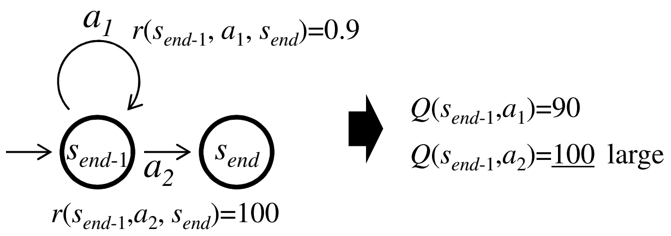
$$r(s,a,s') = \begin{cases} r_{tmp}(s,a,s') & (s'\text{is not terminal}) \\ \dfrac{r_{tmp}(s,a,s')}{1-\gamma} & (s'\text{is terminal}) \end{cases} \tag{6}$$

Here, figure 1(a). shows an example of action value $Q(s,a)$ when the terminal state is not taken into account. The action value is the expected future return value when action a is taken in state s. In this example, when action a_1 is selected in the state immediately before the terminal state, the same state is maintained and the reward 0.9



a_1
$r(s_{end\text{-}1}, a_1, s_{end})$=0.9
$s_{end\text{-}1}$
s_{end}
a_2
$r(s_{end\text{-}1}, a_2, s_{end})$=1.0
$Q(s_{end\text{-}1},a_1)$=90 large
$Q(s_{end\text{-}1},a_2)$=1.0

(a) Action value when terminal state is not considered



a_1
$r(s_{end\text{-}1}, a_1, s_{end})$=0.9
$s_{end\text{-}1}$
s_{end}
a_2
$r(s_{end\text{-}1},a_2, s_{end})$=100
$Q(s_{end\text{-}1},a_1)$=90
$Q(s_{end\text{-}1},a_2)$=100 large

(b) Action value when terminal state is considered

Fig. 1. Confirming effect of the equation (6) by calculating action values

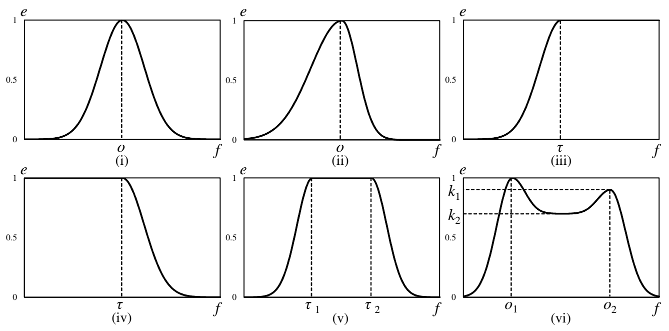


Fig. 2. Evaluation function for each purpose

is received, and when action a_2 is selected the state transitions to the terminal state (goal) and the reward 1.0 is received. The return when $\gamma = 0.99$ is expressed by equation (6), so when action a_1 is continuously selected for infinity, $Q(s_{end-1}, a_1) = 0.9/(1 - 0.99) = 90$. The return when action a_2 is selected is the same as the final reward, so $Q(s_{end-1}, a_2) = 1$. This means that the value of action a_1 is higher, even though the reward is higher for action a_2, which results in a policy that does not move toward the goal, and instead continues to take action a_1. Therefore, the reward when the terminal state is reached is set to the return when a reward of 1.0 is obtained for infinity, that is to say $1.0/(1 - 0.99) = 100$, as shown in figure 1(b). This increases the value of action a_2 and enables to obtain a policy that moves toward the goal.

The six different evaluation functions shown in figure 2. are proposed in accordance with the demands on the evaluation item f.

(i) Evaluation function f has a target value

$$e_k(f) = \exp(-(f - o)^2/\alpha) \tag{7}$$

o is the target value, and α is a parameter that adjusts the tolerance for the offset from the target value.

(ii) The tolerance changes beyond the target value

$$e_k(f) = \begin{cases} \exp(-(f - o)^2/\alpha_1) & (f < o) \\ \exp(-(f - o)^2/\alpha_2) & (f \geq o) \end{cases} \tag{8}$$

α_1 and α_2 are parameters that adjust the respective tolerance for offset in the directions below and above the target value.

(iii) Tolerating offset above the threshold value

$$e_k(f) = \begin{cases} \exp(-(f - \tau)^2/\alpha) & (f < \tau) \\ 1 & (f \geq \tau) \end{cases} \tag{9}$$

τ is the threshold value at which the reward starts to decrease, and α is a parameter that adjusts the tolerance for the offset below the target value. When there is a true lower limit value that cannot be tolerated, set a threshold value that is sufficiently larger than the lower limit value.

(iv) Tolerating offset below the threshold value

$$e_k(f) = \begin{cases} 1 & (f < \tau) \\ \exp(-(f - \tau)^2/\alpha) & (f \geq \tau) \end{cases} \tag{10}$$

τ is the threshold value at which the reward starts to decrease, and α is a parameter that adjusts the tolerance for the offset above the target value. This function is the inverse of equation (9).

(v) There are upper and lower threshold values

$$e_k(f) = \begin{cases} \exp(-(f - \tau_1)^2/\alpha_1) & (f < \tau_1) \\ 1 & (\tau_1 \leq f < \tau_2) \\ \exp(-(f - \tau_2)^2/\alpha_2) & (f \geq \tau_2) \end{cases} \tag{11}$$

τ_1 and τ_2 are the respective upper and lower threshold values at which the reward starts to decrease, and α_1 and α_2 are parameters that adjust the respective tolerance for offset in the directions below τ_1 and above τ_2. This function is a combination of equation (9) and (10).

(vi) There are two target values

$$e_k(f) = \begin{cases} \exp(-(f - o_1)^2/\alpha_1) \\ \qquad (f < o_1) \\ (1 - k2)\exp(-(f - o_1)^2/\alpha_2) + k2 \\ \qquad (o_1 \leq f < \dfrac{o_1 + o_2}{2}) \\ (k1 - k2)\exp(-(f - o_2)^2/\alpha_3) + k2 \\ \qquad (\dfrac{o_1 + o_2}{2} \leq f < o_2) \\ k_1 \exp(-(f - o_2)^2/\alpha_4) \\ \qquad (f \geq o_2) \end{cases} \tag{12}$$

o_1 is the 1st target value and o_2 is the 2nd target value. α_1, α_2, α_3, and α_4 are the respective tolerances for the offset from the target value in each section. k_1 is the 2nd target value priority relative to the 1st target value. k_2 is the parameter that determines how easily the state transitions between target values.

In the proposed scheme, the reward is expressed by the product of all evaluation functions, so new evaluation functions can easily be added. In addition, each evaluation function can set the tolerance for the offset from the target value, so the evaluation item priority can be changed. (For example, to give a certain evaluation item priority, reduce the tolerance of that evaluation function.).

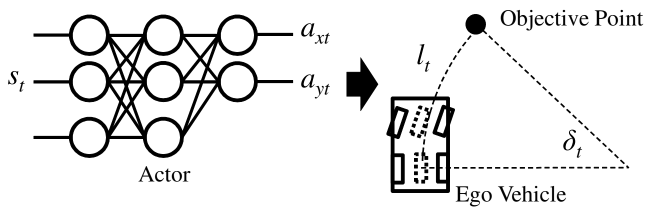


Fig. 3. Outline of the control scheme using the generated next objective point

Table 1. Input from the modded TORCS

Items	Description
Angle	0 is along the track.
Velocity	v_x and v_y in the local coord.
Track position	Distance from the center of the track.
Track	Distance to edge of the track(Each 20 degree).
Others' positon	Near 10 cars' (x, y) in local coord.
Others' speed	Near 10 cars' relative speed.

4. APPLICATION TO AUTOMATED DRIVING

4.1 Test Environment

This chapter describes the application of the proposed scheme to circuit driving and highway cruising to investigate application in regards to automated driving. We use TORCS that is freeware racing simulation with a patch for robot racing and the detail is described in Loiacono et al. (2013). Furthermore, we modify the patch to acquire the positions and relative speeds of ten nearby vehicles within 200m. Table 1 shows the inputs from TORCS.

Figure 3 shows an outline of vehicle control using RL. The actor outputs the acceleration in the local coordinate system, and the controller calculates the target coordinates 0.1s in the future assuming uniform acceleration motion. In addition, the steering angle needed to track the target coordinates is calculated using a 2-wheel model, and the accelerator and brake pedal operations are calculated by PID control. Separation of target coordinate generation and vehicle control enables the controller to absorb differences between vehicle models, so there is no need for relearning depending on the model.

11 courses are used for circuit driving. Learning was performed asynchronously on 10 of the 11 courses to increase the generalization performance. In addition, another course was not used for learning, and was only used as the test course. Circuit driving assumed the condition without other vehicles.

An oval course with a total length of 1770m and models of other vehicles were created to simulate highway cruising. The course has two lanes, a traveling lane and a passing lane, and the other vehicles continuously cruise in the same lane at a velocity of 90 km/h in the traveling lane and 130 km/h in the passing lane.

4.2 Reward Function

The evaluation functions for circuit driving are as follows (figure 4).

$$\begin{aligned}
e_0 &= \exp(-(v_{xt}-250)^2/20000)\\
e_1 &= \begin{cases}\exp(-(p_t+0.5)^2/0.2) & (p_t < -0.5)\\ 1 & (-0.5 \leq f_1 < 0.5)\\ \exp(-(p_t-0.5)^2/0.2) & (p_t \geq 0.5)\end{cases}\\
e_2 &= \begin{cases}\exp(-(a_{xt}+30)^2/20) & (a_{xt} < -30)\\ 1 & (-30 \leq f_2 < 10)\\ \exp(-(a_{xt}+10)^2/20) & (a_{xt} \geq 10)\end{cases}\\
e_3 &= \begin{cases}\exp(-(a_{yt}+40)^2/20) & (a_{yt} < -40)\\ 1 & (-40 \leq a_{yt} < 40)\\ \exp(-(a_{yt}-40)^2/20) & (a_{yt} \geq 40)\end{cases}
\end{aligned} \tag{13}$$

Here, v_{xt} is the velocity [km/h] in the vehicle traveling direction, p_t is the distance from the center of the course (-1 is the left edge and +1 is the right edge), and a_{xt} and a_{yt} are the respective acceleration [m/s2] in the traveling direction and the lateral direction output by the actor. Therefore, the reward is higher when the velocity is close to 250 km/h, the vehicle does not deviate too far from the center of the track, and the output acceleration is not too large.

For highway cruising, an evaluation item as follows is added for avoiding risk with other cars.

$$risk_t = \sum_{k=0}^{n-1} \exp(x_{kt}^2/400 + y_{kt}^2/3) \tag{14}$$

The evaluation functions is as follows (figure 5).

$$\begin{aligned}
e_0 &= \begin{cases}\exp(-(v_{xt}-110)^2/2000) & (v_{xt} < 110)\\ \exp(-(v_{xt}-110)^2/1000) & (v_{xt} \geq 110)\end{cases}\\
e_1 &= \begin{cases}\exp(-(p_t+0.5)^2/0.05) & (p_t < -0.5)\\ 0.1\exp(-(p_t 0.5)^2/0.04)+0.9 & (-0.5 \leq p_t < 0)\\ 0.05\exp(-(p_t-0.5)^2/0.04)+0.9 & (0 \leq p_t < 0.5)\\ 0.95\exp(-(p_t-0.5)^2/0.05) & (p_t \geq 0.5)\end{cases}\\
e_2 &= \begin{cases}\exp(-(a_{xt}+1)^2/30) & (a_{xt} < -1)\\ 1 & (- \leq f_2 < 1)\\ \exp(-(a_{xt}-1)^2/30) & (a_{xt} \geq 1)\end{cases}\\
e_3 &= \begin{cases}\exp(-(a_{yt}+5)^2/30) & (a_{yt} < -5)\\ 1 & (-5 \leq f_3 < 5)\\ \exp(-(a_{yt}-5)^2/30) & (a_{yt} \geq 5)\end{cases}\\
e_4 &= \exp(-(risk_t)^2/0.1)
\end{aligned} \tag{15}$$

Figure 5 also shows the evaluation functions and the $risk_t$ for the case when there is one other vehicle. The reward increases when the velocity is close to 110 km/h, the output acceleration is not too large, and the distance from the other vehicle increases. When there is no vehicle in front, the highest reward is obtained by driving in the left lane at a constant velocity of 110 km/h. However, when driving continuously at 110 km/h, the ego vehicle catches up to the other vehicle driving at 90 km/h. At this time, the reward is higher when the ego vehicle drives at 110 km/h in the right lane than when decelerating to 90 km/h in the left lane.

4.3 Learning Results of Circuit Driving

Figure 6 shows the average return for each 1000 epochs in order to confirm learning convergence. Convergence was

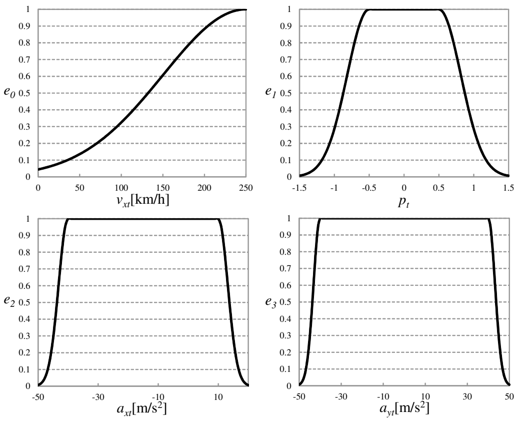


Fig. 4. Evaluation functions for circuit driving

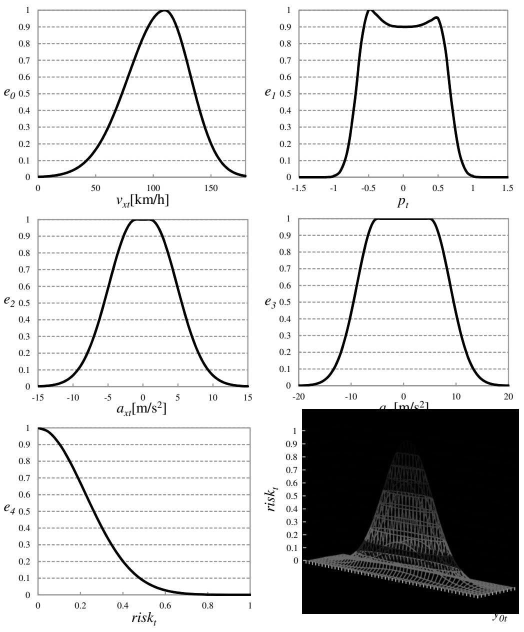


Fig. 5. Evaluation functions for highway cruising

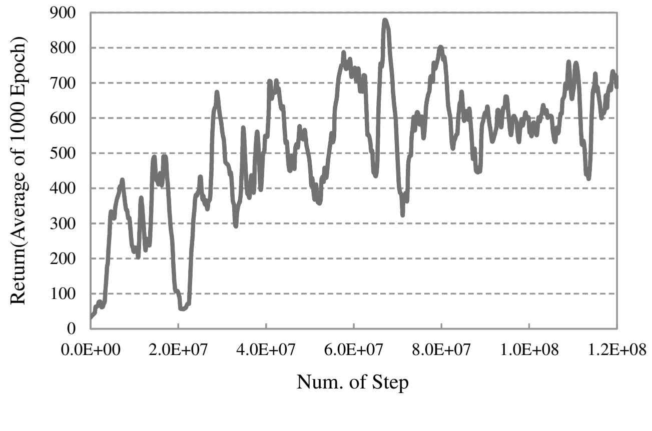


Fig. 6. Return during training in circuit driving

judged and learning ended at 1.2×10^8 steps. This number of steps corresponds to 139 days of time in the simulation.

Figure 7 shows the trajectory (0.5s interval) through an S-shaped curve and hairpin curve when driving on the test course using the obtained policy. This shows that an appropriate policy could be obtained, such as decelerating before the curves and driving the shortest distance through the S-shaped curve.

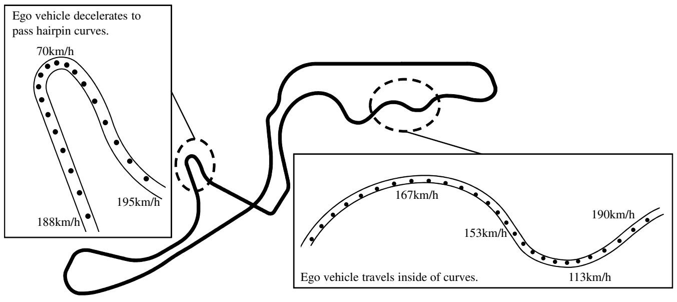

Ego vehicle decelerates to pass hairpin curves.
70km/h
195km/h
188km/h
167km/h
190km/h
153km/h
Ego vehicle travels inside of curves.
113km/h

Fig. 7. Result of trajectory when driving circuit

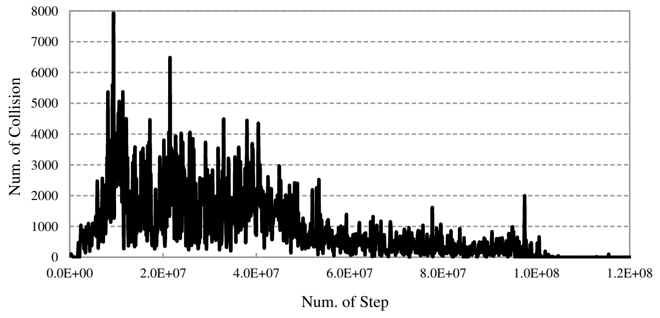


Fig. 8. Collision times during training in highway cruising

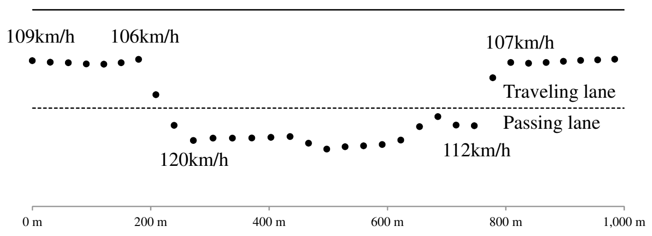


Fig. 9. Result of trajectory when overtaking

4.4 Learning Results for Highway Cruising

Figure 8 shows the number of collisions per 1000s of game time in order to confirm learning convergence. Convergence was judged and learning ended at 1.2×10^8 steps same as the previous experiment.

Next, Figure 9 shows the passing track (1s interval) when driving with the obtained policy. Lane change started when the ego vehicle closed to approximately 30m from the vehicle in front. The reason of the deceleration in the left lane is thought to be avoiding catching up to the front vehicle. Conversely, the reason for acceleration to 110km/h or more in the right lane is that a rear vehicle driving at 130km/h does not catch up to the ego vehicle. The vehicle behavior in the passing lane exhibits some unstable motion. The reason for this is thought to be that the reward between the traveling lane and the passing lane in evaluation function e_1 is too high. Therefore, smoother driving might be obtained by enhancing the evaluation function parameters.

5. CONCLUSION AND FUTURE WORK

This paper proposed a reward function design scheme for automated driving. The proposed reward function evaluates driving states and actions on the process to the destination. Moreover, the scheme can add multiple evaluation items to the reward function. In automated driving, various items should be evaluated, such as velocity, passengers' comfort, distance from obstacles, and so on. Therefore, the scheme is suited for automated driving.

The proposed scheme was also applied to circuit driving and highway cruising to investigate application to automated driving. The results showed that in circuit driving, appropriate driving positions were obtained, such as traveling on the inside of corners, and rapid deceleration to turn along sharp curves was also learned. In highway cruising, lane change in an environment where other vehicles are present, deceleration to avoid catching up to a vehicle in front, and acceleration so that a vehicle to the rear does not catch up to the ego vehicle were learned.

As improving algorithm, risk sensitive reinforcement learning developed by Mihatsch and Neuneier (2002) will be introduced for more safety. In addition, it is also important issue to adjust parameters in the reward function. We are planning to apply inverse reinforcement learning(IRL) cf. Ng et al. (2000); Abbeel et al. (2010) that can estimate reward function using expert action data. We consider that the reward function which is related to safety should be hand-crafted, but the one related to comfortability can be estimated automatically with IRL by Kishikawa and Arai (2019). Therefore, we will develop the scheme combining the hand-crafted reward and the generated reward by IRL.

REFERENCES

Abbeel, P., Coates, A., and Ng, A.Y. (2010). Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13), 1608–1639.

Desjardins, C. and Chaib-Draa, B. (2011). Cooperative adaptive cruise control: A reinforcement learning approach. *IEEE Transactions on intelligent transportation systems*, 12(4), 1248–1260.

Kishikawa, D. and Arai, S. (2019). Comfortable driving by using deep inverse reinforcement learning. In *2019 IEEE International Conference on Agents (ICA)*, 38–43. IEEE.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Loiacono, D., Cardamone, L., and Lanzi, P.L. (2013). Simulated car racing championship: Competition software manual. *arXiv preprint arXiv:1304.1672*.

Michels, J., Saxena, A., and Ng, A.Y. (2005). High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22nd international conference on Machine learning*, 593–600.

Mihatsch, O. and Neuneier, R. (2002). Risk-sensitive reinforcement learning. *Machine learning*, 49(2), 267–290.

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937. PMLR.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

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. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540), 529–533.

Ng, A.Y., Russell, S., et al. (2000). Algorithms for inverse reinforcement learning. In *Icml*, volume 1, 2.

Perot, E., Jaritz, M., Toromanoff, M., and De Charette, R. (2017). End-to-end driving in a realistic racing game with deep reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 3–4.

Schester, L. and Ortiz, L.E. (2021). Automated driving highway traffic merging using deep multi-agent reinforcement learning in continuous state-action spaces. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, 280–287. IEEE.

Stafylopatis, A. and Blekas, K. (1998). Autonomous vehicle navigation using evolutionary reinforcement learning. *European Journal of Operational Research*, 108(2), 306–318.

Ye, F., Cheng, X., Wang, P., Chan, C.Y., and Zhang, J. (2020). Automated lane change strategy using proximal policy optimization-based deep reinforcement learning. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, 1746–1752. IEEE.

Appendix A. LEARNING PARAMETERS

Table A.1 shows the neural network configuration and learning parameters. The neural network has four fully linked layers consisting of an input layer, two intermediate layers, and an output layer. A Rectified Linear Unit (ReLU) function is inserted between the input layer and intermediate layers, and between the intermediate layers. The critic output layer outputs the product-sum operation results as is. The actor output layer outputs the mean and variance for each action. The mean side inserts tan-1 and outputs a value between -1 and 1. The variance side inserts a softmax function and outputs a value of 0 or more. In addition, an offset is added to the variance to promote exploration.

Table A.1. Learning parameters

Items	Values
Hidden layer	500×500
Variance Offset	0.04
Discount Rate	0.99
t_{max}	40
Optimizer	SharedRMSProp
Learning Rate(Actor)	0.0001
Learning Rate(Critic)	0.00005