Backing Up Control of a Self-Driving Truck-Trailer Vehicle with Deep Reinforcement Learning and Fuzzy Logic

Eduardo Bejar and Antonio Morán
Engineering Department
Pontifical Catholic University of Peru, Lima, Peru
ebejar@pucp.pe, amoran@pucp.edu.pe

***Abstract*—This paper presents a control strategy for autonomous truck-trailer systems based on deep reinforcement learning (Deep RL). The deep deterministic policy gradient (DDPG) is used for training the controller using a reward function defined in terms of the desired final state of the system. A fuzzy-logic approach is employed to avoid the truck-trailer jackknife state. Simulation results show that the designed controller exhibits similar performance to state-of-the-art controllers such as the linear-fuzzy controller but with a much simpler design process.**

***Index Terms*—reinforcement learning, deep learning, self-driving vehicles, control systems.**

I. Introduction

Automated truck-trailer vehicles are important to the logistic industry since they allow reduced costs of transportation and fuel consumption while reducing emission levels. Although these vehicles are far from being the norm in the short term, they have been widely used in warehouses and ports in developed countries [1], [2]. These systems are subject to different control goals such as robot positioning, linear and nonlinear trajectory following, backing up, path planning, parallel parking, jackknife avoidance, among others [3].

In particular, backing up of truck-and-trailers to a loading dock is one the most difficult control objectives in mobile robotics [1]. This system is nonlinear, nonholonomic, unstable and underactuated which complicates the controller design [1], [3], [4], [5]. Moreover, the so-called jackknife state, that occurs when the truck and trailer form a right angle, makes the control task even harder. For example, if the truck-trailer reaches this state, the control may be impossible since the steering angle has little effect in the trailer back [6].

Control of truck-trailers has been widely studied in the past years resulting in several aproaches for these systems based mainly on fuzzy-logic, neural networks and nonlinear control. To make the truck-trailer park in a prescribed parking lot [4] and [5] implemented fuzzy controllers that are based on experience from a human-driver translated in linguistic rules. Likewise, in [7] and [8], neural networks were used to learn control policies based on static and dynamic learning algorithms. Moreover, other approaches, as [9], merged neural networks and fuzzy logic. In [1], a Lyapunov-based approach was used to achieve line-path following control. More recently, [3] combined fuzzy logic and linear controllers for path following of linear, circular and sinudoidal trajectories. More techniques are thoroughly described in [10].

In the meanwhile, deep reinforcement learning (Deep RL) has gained a lot of attention in the last years because it has been able to achieve amazing success such as beating the world Go champion [11] and learning to play Atari games [12], [13]. In addition, Deep RL is also solving difficult tasks in robotics such as grasping and manipulation with robotic arms [14], [15]. There are also increasing efforts to apply Deep RL in process control, a field that has been dominated by human-engineered PID controllers for decades [16]. For example, Deep

RL algorithms based on actor-critc methods ([17], [18]) have been applied in [19] in paper manufacturing machines and high-distillation columns, whilst in [20] and [21], these techniques were applied to heating and air conditioning problems.

Deep RL combines the power of deep neural networks that are able to learn representations of complex data and reinforcement learning that aims to learn control policies of agents interacting with an environment. In particular, Deep RL has been applied to truck-and-trailers for speed control and lane change decision making [22]. In [23], a reinforcement learning strategy based on temporal difference learning ([24], Chapter 6) was used for backing up a truck-trailer, however, it did not show promising results since the controller required more than $20,000$ iterations to reach a satisfactory behavior.

This paper is concerned with the backing up control of a truck-and-trailer vehicle with a deep RL technique. Specifically, the paper proposes a neuro-fuzzy controller based on the DDPG algorithm. A reward function is chosen such that it reflects the desired final state of the system. To avoid the jackknife state, a fuzzy-logic approach is implemented in the control action. The effectiveness of the proposed technique is tested on two experiments from the simulation environment presented in [3]: parking in a prescribed parking spot and path-following of a straight line. The paper is structured as follows: Section II presents the model of the truck-and-trailer system. Section III deals with the proposed control strategy that is based on a neuro-fuzzy controller trained with a reinforcement learning approach. Section IV shows the simulation results. Finally, the paper ends in section V with conclusions.

II. Problem Formulation

A. System Model

This paper uses the truck-and-trailer system presented in [3] depicted in Fig. 2. With the aid of Fig. 1, the dynamics of the truck-and-trailer are given by Eq. (1) when the following assumptions are met: the robot moves at low speeds such that its wheels do not side-slip and the linear velocities of traction and rear wheels of the truck and trailer are aligned.

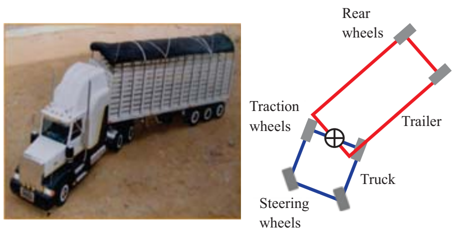

Fig. 1: Truck-and-trailer vehicle [3].

$$\begin{aligned}\dot{x} &= v\cos(\theta_{12})\cos(\theta_2)\\ \dot{y} &= v\cos(\theta_{12})\sin(\theta_2)\\ \dot{\theta}_1 &= -\frac{v}{L_1}\tan(\delta)\\ \dot{\theta}_2 &= -\frac{v}{L_2}\sin(\theta_{12})\end{aligned} \tag{1}$$

In Eq. (1), x and y are the coordinates of the truck and trailer rear wheels in the X-Y plane, θ_1 and θ_2 are the angles of the truck and trailer respect to the X-axis, θ_{12} is the angle of the truck respect to the trailer, δ is the steering angle, L_1 and L_2 are the lengths of the truck and trailer, respectively, and v is the backward speed of the truck-trailer. Moreover, since $\theta_{12} = \theta_1 - \theta_2$, it follows from Eq. (1) that:

$$\dot{\theta}_{12} = \frac{v}{L_2}\sin(\theta_{12}) - \frac{v}{L_1}\tan(\delta) \tag{2}$$

B. Control Objective

This paper is concerned with the backing up control of a truck-and-trailer in two scenarios. First, when the mobile robot has to move to a specific parking spot from a given location as shown in Fig. 3. In particular, for simplicity, only the case when the parking spot is located in the Y axis will be analyzed. Second, when the control objective is to follow a straight line starting from a random location as shown in Fig. 4.

III. Proposed Method

A. Preliminaries

The controller design starts by choosing an appropiate model of the system. Section II, presented

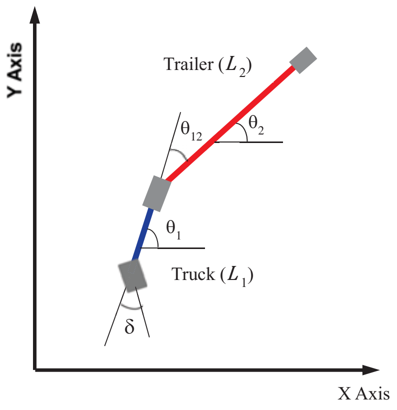

Fig. 2: Truck-and-trailer modelled as two articulated bars [3].

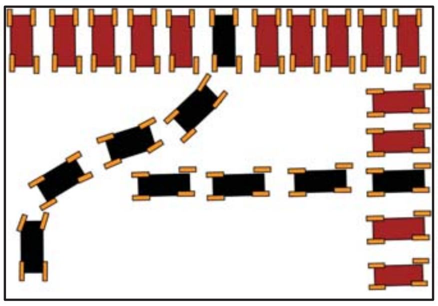

Fig. 3: Mobile robot moving to specific parking spot.

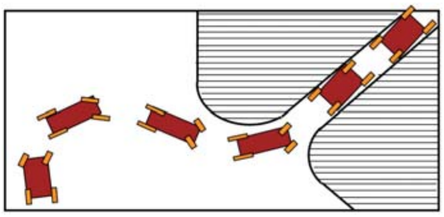

Fig. 4: Mobile robot following a straight line.

Eq. (1) which accuretely describes the system. However, it is more convenient to use θ_{12} instead of θ_1 in the design process [3], resulting in the state-space model shown in Eq. (3):

$$\begin{aligned} \dot{x} &= v\cos(\theta_{12})\cos(\theta_2) \\ \dot{y} &= v\cos(\theta_{12})\sin(\theta_2) \\ \dot{\theta}_2 &= -\frac{v}{L_2}\sin(\theta_{12}) \\ \dot{\theta}_{12} &= \frac{v}{L_2}\sin(\theta_{12}) - \frac{v}{L_1}\tan(\delta) \end{aligned} \tag{3}$$

In addition, since the system is nonholonomic, asymptotic stability cannot be reached on x and y at the same time. Hence, the state vector will be defined as [3]:

$$s = \begin{bmatrix} y \\ \theta_2 \\ \theta_{12} \end{bmatrix} \tag{4}$$

B. Deep RL: Actor-Critic

The goal in reinforcement learning is to maximize the following cumulative reward function:

$$R_t = r_t + \gamma r_{t+1} + ... + \gamma^T r_{t+T} \tag{5}$$

where γ is a discount factor that ranges from 0 to 1. Additionally, the control actions are chosen such that:

$$a_t^* = \text{argmax}_{a_t} Q(s_t, a_t) \tag{6}$$

where s_t and a_t are the state of the system and the control action at timestep t. Moreover, the control actions are taken from a control policy μ, i.e., $a_t = \mu(s_t)$. Likewise, $Q(s_t, a_t)$ is the action-value function defined as:

$$Q(s_t, a_t) = \mathbb{E}[R_t|s_t, a_t] \tag{7}$$

In particular, to maximize Eq. (5), it is important to find the optimal control policy μ' that leads to the optimal action-value function [24]:

$$Q^*(s_t, a_t) = \max_\mu \mathbb{E}[R_t|s_t, a_t] \tag{8}$$

Equation (8) can be solved iteratively with the Bellman equation:

$$Q^*(s_t, a_t) = \mathbb{E}[r(s_t, a_t) + \gamma Q^*(s_{t+1}, a_{t+1})] \tag{9}$$

Optimization of Eqs. (5) and (6) is done with an actor-critic approach [12], [17], [18]. Within this framework, two deep neural networks are used to compute the optimal control actions a_t^* and action-value function $Q^*(s_t, a_t)$, which are called actor and critic networks, respectively. The actor network is represented by $\mu(s_t, \theta_a)$ and the critic network by $Q(s_t, a_t, \theta_c)$ where θ_a and θ_c are the parameters of the actor and critic networks, respectively. The critic network is trained in a supervised learning manner with gradient descent to predict the optimal action-value function given by Eq. (9). Therefore, the cost function that this network minimizes is:

$$J(\theta_c) = \mathbb{E}[(Q^*(s_t, a_t) - Q(s_t, a_t, \theta_c))^2] \quad (10)$$

where $Q^*(s_t, a_t)$ is computed with Eq. (9). The gradient of this cost funcion is given by:

$$\frac{\partial J(\theta_c)}{\partial \theta_c} = \mathbb{E}[Q^*(s_t, a_t) - Q(s_t, a_t, \theta_c)) \frac{\partial Q}{\partial \theta_c}] \quad (11)$$

At the same time, the optimal control actions a_t^*, given by Eq. (6), are computed with gradient ascent, where the gradient of the actor network is given by:

$$\frac{\partial Q(s_t, a_t, \theta_a)}{\partial \theta_a} = \mathbb{E}[\frac{\partial Q(s_t, a_t, \theta_a)}{\partial a_t} \frac{\partial a_t}{\partial \theta_a}] \quad (12)$$

Equation (11) shows that $\partial J(\theta_c)/\partial \theta_c$ requires the computation of $Q^*(s_t, a_t)$, which in turn depends on $Q^*(s_{t+1}, a_{t+1})$. As in [12], a target critic network Q' is used to compute Q^* at s_{t+1} and a_{t+1}. Moreover, a_{t+1} is computed with a target actor network. The target actor and critic networks are two additional neural networks with parameters θ'_a and θ'_c, respectively, with update laws: $\theta'_a \leftarrow \tau\theta_a + (1-\tau)\theta'_a$ and $\theta'_c \leftarrow \tau\theta_c + (1-\tau)\theta'_c$, where $\tau \ll 1$ [18].

C. Learning algorithm

The controller is trained with the DDPG algorithm, which has been studied extensively in [12], [18], [17], [19], etc. In this work, this algorithm was modified to include the reward function from Eq. (13). The algorithm includes a replay memory to estimate the expectations of Eqs. (11) and (12). It also includes actor and critic target networks to compute $Q^*(s_{t+1}, a_{t+1})$ and a_{t+1} from Eq. (9). In addition, the algortithm also uses an explotary noise from the Ornstein-Uhlenbeck process [25] to explore new states.

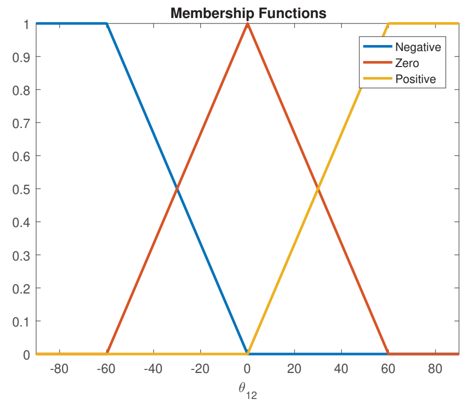

Fig. 5: Membership functions.

Finally, in this work, the following reward function is used [19]:

$$r_t(s_t, a_t, s_{t+1}) = \begin{cases} c, \text{if} \mid s^i - s^{*i} \mid \leq \epsilon, \forall i \\ -\|s - s^*\|_1, \text{otherwise} \end{cases} \quad (13)$$

where $s^* = \begin{bmatrix} y^* & \theta_{12}^* & \theta_2^* \end{bmatrix}^T$ represents the final goal state or setpoint at steady state, and c is a design parameter.

D. Fuzzy-logic

A fuzzy-logic approach is used in order to avoid the jackknife state. Figure 5 shows the membership functions applied to θ_{12}. The fuzzy rules are summarized in Eq. (14):

$$\begin{aligned} \text{IF} \quad \theta_{12} = \text{Positive} \quad \text{THEN} \quad \delta = \delta_{max} \\ \text{IF} \quad \theta_{12} = \text{Zero} \quad \text{THEN} \quad \delta = \delta_N \\ \text{IF} \quad \theta_{12} = \text{Negative} \quad \text{THEN} \quad \delta = \delta_{min} \end{aligned} \quad (14)$$

where "Positive", "Zero" and "Negative" correspond to the regions shown in Fig. 5, δ_{min} and δ_{max} are the minimum and maximum values of the steering angle. In addition, δ_N corresponds to the control actions from the learned controller.

IV. Simulation Results

The controller was trained with the DDPG algorithm with the considerations from Section III. The final goal state was set to $s^* = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$. To reach different setpoints, the control action is

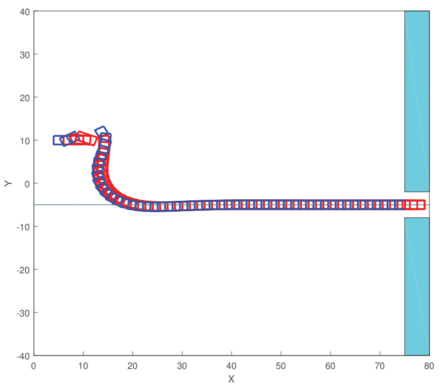

Fig. 6: Parking in a perpendicular parking spot. Initial position: $x = 10$, $y = 10$, $\theta_1 = 0$, $\theta_{12} = 0$.

modified to: $\delta = \mu(s_t - s^*) + \delta^*$, where δ^* is the solution of equation (3) at steady state. Moreover, the steering angle is bounded to $\delta_{min} = -30°$ and $\delta_{max} = 30°$. Several setpoints were randomly chosen from $\{-3\pi/4, -\pi/2 - \pi/4, 0, \pi/4, \pi/2, 3\pi/4\}$ for θ_{12} and θ_2. Figures 6 and 7 show the behavior of the controller when it attempts to attain a prescribed parking lot set at $y = -5$ and $y = 0$, respectively. Note that in Fig. 7 the controller is able to attain the control objective even when starting from the jackknife state. Figures 8 and 9 show the controller performance in the path-following task. The trajectory was set to be a straight line with an inclination of 45 degrees. In this scenario, $y^* = x - 40$ as indicated in [3]. The controller has success at this control task even when coming from the jackknife state.

V. Conclusions

A fuzzy-neuro controller was designed for backing up control of a truck-and-trailer vehicle. A learning algorithm based on the DDPG algorithm was used for training the neuro-controller. A fuzzy logic approach was employed to avoid the jackknife state. The controller was able to meet the control objectives of parking into a perpendicular parking spot and path-following of a straight line even when starting from the jackknife state. Simulation results showed that the controller performance was similar to [3], however, the controller design was simpler because it did not require a linearization procedure. Moreover, the controller required less than half of the training episodes needed in [23] to achieve a satisfactory behavior.

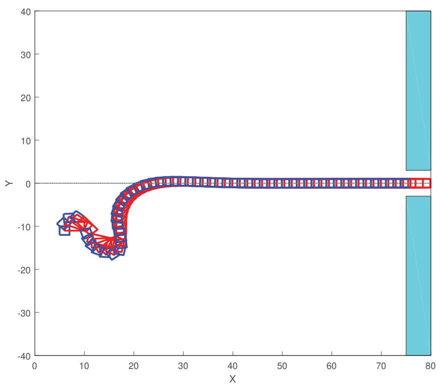

Fig. 7: Parking in a perpendicular parking spot. Initial position: $x = 10$, $y = -10$, $\theta_1 = \pi/2$, $\theta_{12} = \pi/2$.

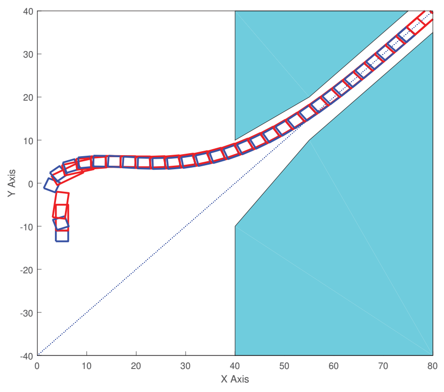

Fig. 8: Line-path following. Initial position: $x = 5$, $y = -5$, $\theta_1 = \pi/2$, $\theta_{12} = 0$.

Acknowlegments

This research was supported in part by the Master's program in Control and Automation Engineering – Graduate School at Pontifical Catholic University of Peru.

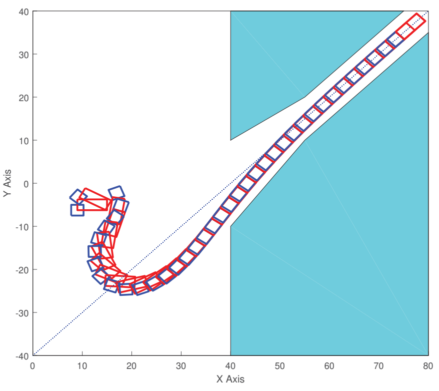

Fig. 9: Line-path following. Initial position: $x = 15$, $y = -5$, $\theta_1 = \pi/2$, $\theta_{12} = \pi/2$.

References

[1] A. Widyotriatmo, P. Siregar, and Y. Nazaruddin, "Line following control of an autonomous truck-trailer," *Proceedings of the 2017 International conference on Robotics, Biomimetics and Intelligent Computational System (Robionetics)*, pp. 24–28, August 2017.

[2] P. Ritzen, E. Roebroek, N. van de Wouw, Z. Jiang, and H. Nijmeijer, "Trailer steering control of a tractortrailer robot," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 4, pp. 1240–1252, July 2016.

[3] A. Moran, "Autonomous path following of truck-trailer vehicles using linear-fuzzy control," *Proceedings of the 3rd International Conference on Control, Automation and Robotics (ICCAR)*, pp. 651–657, April 2017.

[4] S. Kong and B. Kosko, "Backing up a truck-trailer with suboptimal distance trajectories," *Proceedings of the IEEE 5th International Conference on Fuzzy Systems*, pp. 1439–1445, September 1996.

[5] J. Yi, N. Yubazaki, and K. Hirota, "Backing up control of truck-trailer system," *Proceedings of the IEEE 10th International Conference on Fuzzy Systems*, pp. 489–492, December 2001.

[6] M. Ho, P. Chan, A. Rad, and C. Mak, "Truck backing up neural network controller optimized by genetic algorithms," *Proceedings of the 2003 Congress on Evolutionary Computation (CEC)*, pp. 944–950, December 2003.

[7] D. Nguyen and W. B, "The truck backer-upper: An example of self-learning in neural networks," *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, vol. 2, pp. 357–363, 1989.

[8] H. Kinjo, B. Wang, and T. Yamamoto, *Proceedings of the 26th Conference of IEEE Industrial Electronics Society (IECON*.

[9] S. Kong, , and B. Kosko, "Adaptive fuzzy systems for backing up a truck-and-trailer," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 211–223, March 1992.

[10] J. David and P. V. Manivannan, "Control of truck-trailer mobile robots: a survey," *Service Robotics Journal, Springer Verlag*, vol. 7, pp. 245–258, 204.

[11] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. R. Baker, M. Lai, A. Bolton, Y. Chen, T. P. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354–359, 2017.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, 2015.

[14] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller, "Data-efficient deep reinforcement learning for dexterous manipulation," arXiv preprint arXiv:1704.03073, 2017.

[15] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," arXiv preprint arXiv:1806.10293, 2018.

[16] K. J. Astrom and T. Hagglund, "The future of pid control," *Control Engineering Practice*, vol. 9, pp. 1163–1175, November 2001.

[17] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," *International Conference on Learning Representations (ICLR)*, 2014.

[18] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *International Conference on Learning Representations (ICLR)*, 2016.

[19] P. L. S.P.K. Spielberg, R.B. Gopaluni, "Deep reinforcement learning approaches for process control," *Proceedings of the 6th International Symposium on Advanced Control of Industrial Processes (AdCONIP)*, pp. 201–206, May 2017.

[20] Y. Wang, K. Velswamy, and B. Huang, "A long-short term memory recurrent neural network based reinforcement learning controller for office heating ventilation and air conditioning systems," *Processes*, vol. 5, no. 3, 2017.

[21] D. N. N. Amir-massoud Farahmand, Saleh Nabi, "Deep reinforcement learning for partial differential equation control," *Proceedings of the 2017 American Control Conference*, pp. 3120–3127, May 2017.

[22] C.-J. Hoel, K. Wolff, and L. Laine, "Automated speed and lane change decision making using deep reinforcement learning," arXiv:1803.10056.

[23] C. Gatti and M. Embrechts, "An application of the temporal difference algorithm to the truck backer-upper problem," *Proceedings of the European Symposium on Artificial Neural Networks(ESANN)*, pp. 135–140, April 2014.

[24] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*, 2nd ed. MIT Press, 1998.

[25] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Physical Review*, 36(5):823, 1930.