Journal of The Korea Society of Computer and Information
Vol. 23 No. 10, pp. 23-31, October 2018
www.ksci.re.kr
https://doi.org/10.9708/jksci.2018.23.10.023

Controller Learning Method of Self-driving Bicycle Using State-of-the-art Deep Reinforcement Learning Algorithms

Seung-Yoon Choi*, Tuyen Pham Le*, Tae-Choong Chung*

Abstract

Recently, there have been many studies on machine learning. Among them, studies on reinforcement learning are actively worked. In this study, we propose a controller to control bicycle using DDPG (Deep Deterministic Policy Gradient) algorithm which is the latest deep reinforcement learning method. In this paper, we redefine the compensation function of bicycle dynamics and neural network to learn agents. When using the proposed method for data learning and control, it is possible to perform the function of not allowing the bicycle to fall over and reach the further given destination unlike the existing method. For the performance evaluation, we have experimented that the proposed algorithm works in various environments such as fixed speed, random, target point, and not determined. Finally, as a result, it is confirmed that the proposed algorithm shows better performance than the conventional neural network algorithms NAF and PPO.

▸Keyword: Reinforcement Learning, DDPG, Machine Learning, Bicycle Self Balancing, Deep Learning

I. Introduction

자전거는 환경 친화적이며 저렴하고 사용자에게 유용한 이동수단이다. 그러나 자전거의 불안정한 역동성으로 인해 조향을 하는 사람은 자전거를 제어하기 위해 상당한 노력을 해야 한다. 이미 우리는 자율 운전 자동차, 자율 항공기 등의 이동수단을 개발하고 있는데, 메커니즘과 컨트롤러 양 측면에서 자전거 제어를 향상시키는 방법에 대해 많은 연구가 이루어지고 제안되었음에도 불구하고 [1-4], 무인 운전 자전거는 여전히 현실화 되지 않은 이동수단으로 남아있다[5]. 구체적으로, 자전거의 물리적 향상에 초점을 맞추는 연구들[2,3]과 제어 이론과 자전거 역학에 관한 지식에 기반하여 자전거 컨트롤러를 만들려는 연구들[1,4]이 있다. 그러나, 제안된 컨트롤러는 실험 환경에서만 제대로 작동하고 환경의 장애 요소로 인해 현실 세계에 대하여 적용 하는 데는 실패했다. 주변과 상호작용하는 기능이 가능한 강화 학습 기반 컨트롤러는 다양한 환경에 대하여 적응할 수 있을 것으로 기대할 수 있다[19-21]. 이러한 가능성에도 불구하고, 자전거 제어기를 구축하는데 강화학습을 사용한 연구는 매우 적다[4,6,7]. Randlov[6]는 높은 수준의 연속적 상태와 행위 공간을 다루기에는 충분하지 않은 SARSA 알고리즘을 기반으로 컨트롤러를 구축하였다. Jie Tan[4]은 파라미터들을 학습하기 위하여 정책 경사(Policy Gradient)를 적용한 얕은 수준의 신경망을 사용하였다[8]. 그러나 얕은 신경망 제어기는 자전거와 같은 고도의 비선형 환경을 표현하는데 한계가 있다. Tuyen [7]은 심층 신경망을 사용했는데 그의 구현에서 컨트롤러는 DDPG(Deep Deterministic Policy Gradient) 알고리즘[9]을 사용하여 학습한다. 그리고 자전거는 완전하게 자기 균형을 이룰 수 있었다. 그러나 해당 연구에서 그 컨트롤러는 자전거를 임의로 주어진 위치로 인도 하는데는 실패하였다. 본 연구에서는 자전거를 어느 곳으로나 인도 할 수 있는 개선된 제어기를 제안한다. 제어기는 핸들 바에 가해진 토크와 질량 중심과 자전거 계획 사이의 변위를

• First Author: Seung-Yoon Choi, Corresponding Author: Tae-Choong Chung
*Seung-Yoon Choi (sychoi84@khu.ac.kr), Dept. of Computer Engineering, Kyung Hee University
*Tuyen Pham Le (tuyenple@khu.ac.kr), Dept. of Computer Engineering, Kyung Hee University
*Tae-Choong Chung (tcchung@khu.ac.kr), Dept. of Computer Engineering, Kyung Hee University
• Received: 2018. 09. 17, Revised: 2018. 10. 01, Accepted: 2018. 10. 01.
• The authors are grateful to the Basic Science Research Program through the National Research Foundation of Korea (NRF-2017R1D1A1B04036354).

입력으로 받는다. 변위값이 작으면 높은 속도의 자전거를 다루기 어렵다. 따라서 본 논문에서는 자전거가 어떤 지점에서 시작하던지 모든 장소에 도착할 수 있도록 할 수 있는 제어기를 학습 과정을 개선하는데 초점을 맞춘다.

본 논문의 기여점은 다음과 같다. 첫 번째로 자전거 속도를 제어하기 위하여 자전거 동력학을 재정의한다. 수정된 역학은 동적인 자전거 속도를 다룰 수 있을 것으로 예상된다. 두 번째, 자전거가 균형을 잡도록 조정하는 일 뿐만이 아니라 목적지로 갈 수 도록 보상 기능을 목표로 한 학습 과정을 제안한다. 학습 과정은 DDPG를 기반 알고리즘으로 사용한다.

논문의 나머지 부분은 다음과 같이 구성된다. 2 절에서는 배경 지식에 대해 설명한다. 3절에서는 본 논문에서 제안하는 방법에 대해 소개한다. 4절에서는 제안한 방법의 실험 결과에 대하여 설명한다. 마지막으로 5절에서는 본 논문에 대한 결론을 요약한다.

II. Preliminaries

1. Related works

1.1 Markov Decision Process and Reinforcement Learning

강화학습은 환경에서 직접 받은 보상 값으로부터 학습을 하는데 중점을 둔 기계 학습의 한 종류이다. 강화학습에서는 에이전트를 통해서 환경과 상호작용하며 학습하는데 이때 에이전트는 환경이 주는 정보인 보상을 통해서 어떤 행동을 더 해야 할지 알 수 있고 사전지식이 없어도 학습할 수 있다.

기본적으로 강화학습 문제는 S, A, P, r, γ의 튜플 을 가지는 이산 시간 마르코프 결정 프로세스(Markov Decision Process: MDP)로 모델링 된다. 튜플은 상태 공간 S, 행동 공간 A, 현재 상태의 동작 쌍이 주어지면 다음 상태를 획득할 확률을 측정하는 함수 P, 각 상태-행동 쌍에서 얻은 보상 r을 정의하고, (0,1) 사이의 값을 갖는 γ는 감가율을 표시한다. 상태-행동 쌍의 시퀀스는 다음 수식에 따라 보상 값을 누적하며 에피소드의 궤적을 생성한다.

$$R(\xi) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \quad (1)$$

강화 학습 문제의 정책은 상태 공간을 행동 공간에 매핑하는 함수 π이다. 강화학습 알고리즘은 다음 수식에 따라 기대되는 차감된 보상(Discounted Reward)를 최대화 하는 최적의 정책 π^*를 찾는다.

$$J(\pi) = \mathbb{E}\ [R(\xi)] = \int p(\xi|\pi) R(\xi) d\xi \quad (2)$$

정책 경사 기반 방법 (Policy gradient based method) 은 최적 정책을 찾는 방법 중 하나이다. 정책 경사 기반 방법에서 정책은 매개변수 벡터 θ에 의해 매개변수화 되고, 다음 수식에 따라 기대되는 차감된 보상 값(Discounted Reward Value)의 경사 방향을 따라 업데이트 된다.

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi(\theta_k)) \quad (3)$$

위 수식에서 α는 학습률을 타나내고 k는 현재 업데이트 횟수를 나타낸다.

1.2 Deep Deterministic Policy Gradient Algorithm

DDPG[6]는 정책을 나타내기 위하여 심층 신경망을 사용하는 오프-폴리시(Off-Policy) 알고리즘이다. 이 알고리즘의 특징은 다음과 같다. 첫째, 알고리즘은 액터-크리틱(Actor-Critic) 프레임 워크를 상속받는다[10]. 이것은 두개의 구성 요소가 있음을 의미하는데, 하나는 액터(Actor)이고 다른 하나는 크리틱(Critic)이다. 액터는 상태를 입력 받아 동작을 생성하는 정책을 관리한다. 크리틱은 액터의 가치를 평가하는데 사용되는 행동 가치 함수를 추정한다. 둘째, DDPG알고리즘은 액터와 크리틱 파트 구현을 위하여 두 개의 심층 신경 네트워크를 사용한다. 따라서 DDPG는 자전거를 제어하는 것과 같은 고도의 비선형적인 작업을 표현하기에 충분히 유용하다. 셋째, 이 알고리즘은 액터 네트워크(Actor Network)를 학습하기 위하여 다음 식과 같이 결정론적 정책 경사(Deterministic Policy Gradient)[11] 를 사용한다.

$$\nabla_\theta J(\pi(\theta^\mu)) \approx \mathbb{E}\ \left[\nabla_\alpha Q(s, a|\theta^Q) \times \nabla_{\theta^\mu} \mu(s|\theta^\mu)\right] \quad (4)$$

위 수식에서 $\nabla_\theta J(\pi(\theta^\mu))$는 정책 경사(Policy Gradient)이고, $\nabla_a Q(s, a|\theta^Q)$는 행동(Action) a에 대한 행동 가치 함수의 기울기이다. $\nabla_{\theta^\mu} \mu(s|\theta^\mu)$는 파라미터 θ^μ에 대한 액터의 기울기이다. 마지막으로 DDPG 알고리즘은 딥-큐러닝(Deep Q-Learning)의 두 가지 특성을 사용한다[12]. 첫 번째 특성은, 액터 네트워크의 사본과 크리틱 네트워크의 사본을 유지하는 것이다. 사본은 학습 단계에서 안정성을 향상시킨다. 두 번째 특징은, 환경과 상호작용하는 동안의 모든 샘플 데이터를 저장하는 리플레이 메모리를 유지하는 것이다. 매 스텝마다 재생 메모리에서 데이터 묶음을 무작위로 샘플링하여 네트워크를 학습한다. 재생 메모리를 사용하여 일련의 데이터 샘플들 사이의 상관관계를 제거할 수 있다.

2. Bicycle Dynamics

이륜 자전거 동력학(Bicycle Dynamics)에 대한 연구는 여러 연구자들에 의하여 이루어졌다[1,4,6,13]. 본 연구는 Randlov [6]의 연구를 계승하고 있을 뿐 아니라 자전거의 속도가 동적인 복잡한 경우까지 확장한다. 자전거는 $(\omega, \dot{\omega}, \ddot{\omega}, \theta, dottheta, \psi_g)$여섯 가지 상태 요소를 가지고 있다. 여기에서 $\omega, \dot{\omega}, \ddot{\omega}$ 는 각각 자전거에서 수직면까지의 각도, 각속도 및 각가속도이다. θ,$dottheta$는 핸들바의 각도와 각속도이다. ψ_g는 자전거와 지정된 목표 g가 이루는 각도이다. 자전거를 제어하기 위하여 에이전트는 세 가지 행동을

선택한다. 첫 번째 행동은 핸들 바에 적용되는 토크(T)이다. 두 번째 행동은 Fig. 1에서처럼 무게 중심과 자전거 면 사이의 변위(d)이다. 세 번째 행동은 자전거의 페달에 적용되는 힘(F)이다.

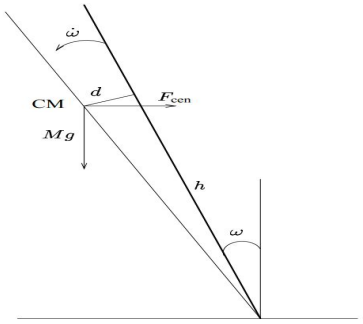

Fig. 1. The bicycle as seen from behind. The thick line represents the bicycle[14]

앞 타이어 용 타이어 위치 방정식은 다음과 같다.

$$\begin{bmatrix} x_f \\ y_f \end{bmatrix}_{(t+1)} = \begin{bmatrix} x_f \\ y_f \end{bmatrix}_{(t)} + vdt \begin{bmatrix} -\sin\left(\psi+\theta+sign(\psi+\theta)\arcsin\left(\frac{vdt}{2r_f}\right)\right) \\ \cos\left(\psi+\theta+sign(\psi+\theta)\arcsin\left(\frac{vdt}{2r_f}\right)\right) \end{bmatrix} \quad (5)$$

그리고 뒷 타이어용은 다음과 같다.

$$\begin{bmatrix} x_b \\ y_b \end{bmatrix}_{(t+1)} = \begin{bmatrix} x_b \\ y_b \end{bmatrix}_{(t)} + vdt \begin{bmatrix} -\sin\left(\psi+\theta+sign(\psi+\theta)\arcsin\left(\frac{vdt}{2r_b}\right)\right) \\ \cos\left(\psi+\theta+sign(\psi+\theta)\arcsin\left(\frac{vdt}{2r_b}\right)\right) \end{bmatrix} \quad (6)$$

여기에서 ψ는 자전거와 수평선에 의해 각도가 만들어진다. r_b와 r_f는 Fig. 2에서 처럼 각각 전방 타이어 및 후방 타이어의 회전반경이다.

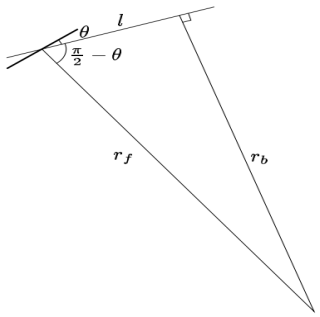

Fig. 2. Seen from above. The thick line represents the front tyre [14]

r_b와 r_f는 각각 다음의 식을 가진다.

$$r_f = \frac{l}{\left|\cos\left(\frac{\pi}{2}-\theta\right)\right|} = \frac{l}{|\sin\theta|} \quad (7)$$

$$r_b = l\left|\tan\left(\frac{\pi}{2}-\theta\right)\right| = \frac{l}{[\tan\theta]} \quad (8)$$

각가속도 $\ddot{\omega}$는 다음과 같이 계산할 수 있다.

$$\ddot{\omega} = \frac{1}{I_{bicycle}}\left(Mhg\sin\phi\left(I_{dc}\dot{\sigma}\dot{\theta}+sign(\theta)v^2\left(\frac{M_dr}{r_f}+\frac{M_dr}{r_b}+\frac{Mh}{r_{CM}}\right)\right)\right) \quad (9)$$

여기에서 각 ϕ는 Fig. 1에서 무게 중심의 전체 경사각이고 다음과 같이 정의된다.

$$\phi = \omega + \arctan\left(\frac{d}{h}\right) \quad (10)$$

앞 타이어와 핸들 바의 각가속도 $ddottheta$는 다음과 같다.

$$ddottheta = \frac{T - I_{dv}dotsigma\dot{\omega}}{I_{dl}} \quad (11)$$

관성 모멘트는 다음과 같은 식을 가진다.

$$I_{bicycle} = \frac{13}{3}M_ch^2 + M_p(h+d_{CM})^2 \quad (12)$$

타이어에 대한 다양한 관성 모멘트는 Fig. 3과 같고 다음과 같이 추정된다.

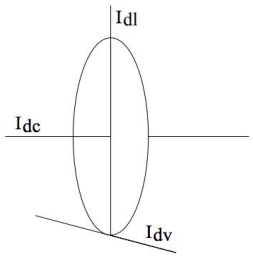

Fig. 3. Axis for moments of Inertia for a tyre [14]

$$I_{dc} = M_cr^2,\ I_{dv} = \frac{3}{2}M_dr^2,\ I_{dl} = \frac{1}{2}M_dr^2 \quad (13)$$

자전거의 속도는 페달에 가해지는 힘을 학습하여 조정할 수 있다. Fig. 4는 페달의 힘이 자전거에 전달되는 방법을 보여준다.

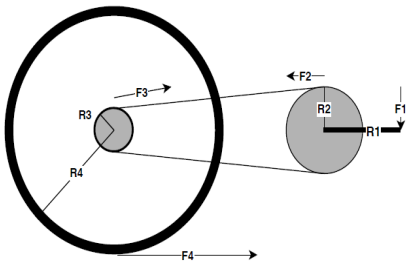

Fig. 4. Force transmission

정적 평형 가정을 사용하여 다음과 같이 토크 방정식을 정의할 수 있다.

$$F_1R_1 = F_2R_2 \quad (14)$$

$$F_3R_3 = F_4R_4 \quad (15)$$

$F2 = F3$이므로, $F4$에 대한 식을 만들기 위하여 위의 두 방정식을 결합할 수 있다.

$$F_4 = F_1\frac{R_1R3}{R_2R_4} \quad (16)$$

F_4는 자전거의 가속도를 결정한다. 그 중에서도 자전거의 가속도는 다음과 같은 식을 갖는다.

$$a = \frac{F_4}{M_c + M_d + M_p} \quad (17)$$

가속도를 통해서 자전거의 속도를 계산할 수 있다.

$$v = v + trianglet \times a \quad (18)$$

자전거 동력학에 대한 다양한 매개 변수는 Table 1과 같다.

Table 1. Parameters of bicycle dynamics

Notation	Description	Value
c	Horizontal distance between the point, where the front wheel touches the ground and the CM	66 cm
CM	The Center of Mass of the bicycle and cyclist as a total	
d	The agent's choice of the displacement of the CM perpendicular to the plan of the bicycle	
d_{CM}	The vertical distance between the CM for the bicycle and for the cyclist	30 cm
h	Height of the CM over the ground	94 cm
l	Distance between the front tyre and the back tyre at the point where they touch the ground	111 cm
M_c	Mass of the bicycle	15 kg
M_d	Mass of a tyre	1.7 kg
M_p	Mass of the cyclist	60 kg
r	Radius of the tyre	34 cm
$dotsigma$	The angular velocity of a tyre	$dotsigm$
T	The torque the agent applies on the handlebars	
dt	time step	$0.025s$

III. The Proposed Scheme

자전거 문제에 강화 학습을 적용하는 방법은 다음과 같다. 자전거의 상태는 $(\omega, \dot{\omega}, \ddot{\omega}, \theta, \dot{\theta}, \psi_g)$로 표현된다. 여기서 $\omega, \dot{\omega}, \ddot{\omega}$는 각각 자전거에서 수직면까지의 각도, 각속도 및 각가속도이고, $\theta, \dot{\theta}$는 핸들 바의 각도와 가속도이며, ψ_g는 자전거와 지정된 목표 g가 이루는 각도이다. 이러한 상태는 각 타임 스텝마다 컨트롤러로 보내지고, 컨트롤러는 d의 값, 토크 T 및 힘 F를 페달에 반환한다. 컨트롤러를 학습하는데 사용되는 DDPG 알고리즘은 Fig. 5에 요약되어 있다. $Q(s,a|\theta^Q)$와 $Q'(s,a|\theta^{Q'})$를 크리틱의 주 네트워크 및 타겟 네트워크이라 하고, $\mu(s,a|\theta^\mu)$와 $\mu'(s,a|\theta^{\mu'})$는 액터의 주 네트워크와 타겟 네트워크 이며, R은 경험 리플레이 라고 하자. 제안하는 방법에서 학습과정은 다음과 같이 표현된다.

Algorithm 1: Deep Deterministic Policy Gradient

Input: Main network and target network of critic $Q(s,a|\theta^Q)$ and $Q'(s,a|\theta^{Q'})$; main nettwork and target network of actor $\mu(s,a|\theta^\mu)$ and $\mu'(s,a|\theta^{\mu'})$; Experience replay R.

Initialize: Randomly initialize Q and μ; Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$ and $\theta^{\mu'} \leftarrow \theta^\mu$; Initialize R

for episode $= 1, 2, ...M$ do

  Initialize: a random process N for action exploration

  Get initial observation state s_1

  for $t = 1,2,...T$ do

    Get $a_t = \mu(s_t|\theta^\mu) + N_t$

    Execute a_t and observe r_t and s_{t+1}

    Store tuple (s_t, a_t, r_t, s_{t+1}) in R

    Sample a batch of N tuples from R

    Compute $y_i = r_i + \gamma Q'(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})|\theta^{Q'})$

    Update critic by minimizing the loss:

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$$

    Update actor by using DPG:

$$\nabla_{\theta^\mu}\mu|_{s_i} \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \times \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end

end

Fig. 5. DDPG Algorithm

자전거와 관련하여 제안한 방법은 다음과 같이 수행된다. 에이전트는 자전거로부터 상태 S_t를 관찰하고 $a_t = \mu(s_t|\theta_\mu + N_t)$에 따라 다음 행동을 얻기 위해 상태를 액터 네트워크에 제공한다. N_t는 행동공간을 탐험하기 위한 작은 불규칙적인 노이즈이다. 자전거는 에이전트에게 다음상태 s_{t+1}과 보상 r_t를 전달한다. 샘플 데이터인 $< s_t, a_t, r_t, s_{t+1} >$은 나중을 위해서 경험 리플레이 메모리에 저장된다. 경험 리플레이 메모리로부터 N개의 샘플의 배치를 무작위로 선택한 후, 그것을 네트워크들의 학습에 사용한다. 이후 손실함수를 최소화하는 방향으로 크리틱 네트워크를 학습한다. 손실함수는 다음과 같이 정의한다.

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2 \quad (19)$$

여기서 $y_i = r_i + \gamma Q'(s_t, \mu'(s_{t+1}|\theta^{\mu'})|\theta^{Q'})$ 이며, Q, μ, Q' 와 μ' 은 크리틱 네트워크, 액터 네트워크 및 타겟 크리틱 네트워크, 타겟 액터 네트워크를 나타낸다. 이후 결정론적 정책 경사(Deterministic Policy Gradient) 정리[11]를 사용하여 액터 네트워크를 학습시킨다. 표현식은 다음과 같다.

$$\nabla_{\theta^\mu}\mu|_{s_i} \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_i,a=u(s_i)} \times \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{si} \quad (20)$$

여기서 $\nabla_a Q$ 는 행동 a에 대한 크리틱의 경사를 나타내고, $\nabla_{\theta^\mu}\mu$은 파라메터 θ^μ에 대한 액터의 경사를 나타낸다. 자전거를 제어하기 위해 보상 함수는 다음과 같이 정의한다.

$$r(s,a) = \begin{cases} the\ last\ reward\ before\ falling\ down & |\omega| > \frac{\pi}{6} \\ -(\omega^2 + 0.1\dot{\omega}^2 + 0.01\ddot{\omega}^2) - 2\psi_g^2 & |\omega| > \frac{\pi}{6} \end{cases} \quad (21)$$

$-(\omega^2 + 0.1\dot{\omega}^2 + 0.01\ddot{\omega}^2)$는 자전거의 균형을 잡는 역할을 하고 $-2\psi_g^2$는 자전거를 목표지점으로 이끌기 위한 것이다. 각 항의 계수는 보상에 대한 기여도를 기준으로 선택된다. 특히 밸런싱을 하는 동안 가장 중요한 ω에 1.0의 계수를 사용한다. 반면 목표지점으로 가는 문제인 goto 목표에서 중요성을 나타내는 ψ_g에는 2.0을 값으로 사용한다.

Fig. 6은 각 에피소드의 마지막 단계에서 보상 함수의 구성 요소 기여도를 보여준다. Fig. 4에서 $-2\psi_g^2$가 보상 함수에서 가장 중요하다는 것을 알 수 있다. 5000 학습 에피소드 후에 구성 요소의 값은 0으로 수렴한다.

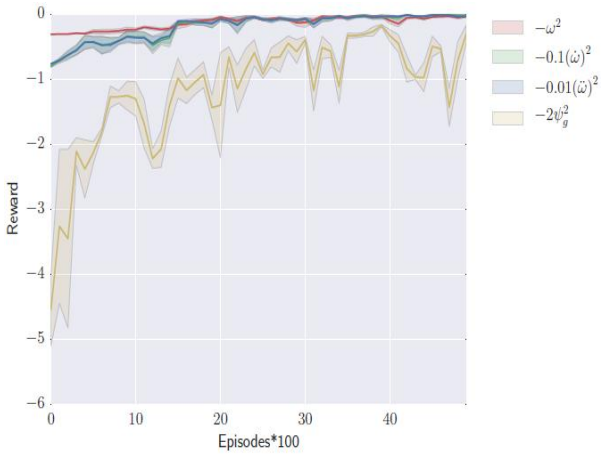

Fig. 6. Contribution of components on reward function

IV. Experiments

실험을 진행한 환경은 다음과 같다. 운영체제는 리눅스(우분투 16.04 LTS)를 사용하였고 AMD FX 8300 옥타코어 프로세서와 32GB DDR3 메모리를 사용하였다. 통합 개발환경으로는 파이참(PyCharm)을 사용하였고 언어는 파이썬을 사용하였다. 실험을 위한 데이터로 사용되는 액터 네트워크와 크리틱 네트워크의 구조는 각각 Table. 1과 Table. 2의 내용과 같다. 본 논문에서는 참고문헌 [10]에 있는 동일한 아키텍처의 액터 네트워크와 크리틱 네트워크를 적용한다. 액터 네트워크의 입력은 상태이고 출력은 동작이다. 크리틱 네트워크의 입력은 행동과 상태의 조합이며 출력은 큐-값(Q-Value)이다. 액터 네트워크의 숨겨진 두개의 레이어는 각각 300개 400개의 유닛을 가지며 크리틱 네트워크의 숨겨진 두개의 레이어는 모두 200개의 유닛을 갖는다. 네트워크 매개변수는 무작위로 초기화되며 ADAM 알고리즘을 사용하여 최적화된다. 타겟 네트워크는 0.001의 학습률로 업데이트 된다. 알고리즘의 매개변수는 Table. 4와 같은데, 행동 공간을 탐색하기 위하여 Ornstein-Uhlenbeck 프로세스[13]를 사용한다. 리플레이 메모리는 500,000 개의 데이터 샘플이 포함되고 각 트레이닝 단계에서 무작위로 64개 샘플의 배치를 가져와서 컨트롤러를 학습하는데 사용한다.

Table 2. Parameters of actor network

Name	Value
Input layer	A state (s_t)
1st fully connected layer	400 units
2nd fully connected layer	300 units
Output layer	An actions $(a_{|t})$
Initial parameters	Uniformly random between $[-3e^{-3}, 3e^{-3}]$
Learning rate	0.001
Optimizer	ADAM[14]

Table 3. Parameters of critic network

Item	Value
Input layer	State-action pair (s_t, a_t)
1st fully connected layer	200 units
2nd fully connected layer	200 units
Output layer	Q-value
Initial parameters	Uniformly random between $[-3e^{-3}, 3e^{-3}]$
Learning rate	0.001
Optimizer	ADAM[14]

Table 4. Parameters of algorithm

Name	Value
Input dimension	6 (states)
Output demension	3 (actions)
Discounted factor	0.99
Random noise	Ornstein-Uhlenbeck Process[21] with θ=0.15 and σ=0.2
Experience memory capacity	500000
Batch size	64 samples

첫 번째 평가에서는 DDPG 알고리즘으로 학습한 자전거의 성능과 다른 알고리즘으로 학습한 자전거의 성능을 비교한다. 그 중에서도 DDPG와 NAF(Normalized Advantage Function) 알고리즘[16] 및 PPO(Proximal Policy Optimization) 알고리즘[17,18]를 비교한다. 두 알고리즘은 고도의 연속적 행위 공간을 처리할 수 있는 심층 강화학습 알고리즘이다. Fig. 7은 50,50 위치에서 무작위로 시작하여 60, 65 m에서 목표위치에 도달하고자 하는 자전거의 성능을 보여준다. 자전거의 속도는 시속 10km로 고정되고 변위는 -20cm에서 20cm 사이 이다. 실험에서는 5000에피소드의 성능을 살펴본다. 각 에피소드에는 400개의 타임스텝이 있다. Fig. 7을 통해서 DDPG에 의해 학습된 제어기가 다른 알고리즘에 의해 학습된 제어기를 능가한다는 것을 알 수 있다. PPO 알고리즘은 점진적으로 안정된 제어기를 학습할 것으로 기대되지만 자전거 도메인에서 DDPG 알고리즘보다 우수하지는 않다. NAF도 자전거에 적합한 컨트롤러를 학습할 수 없었다.

Fig. 7. Compare performance of DDPG with other algorithms on bicycle domain

변위 d가 자전거에 미치는 영향을 보이기 위하여 두 번째 평가에서는 자전거의 성능이 무작위로 50,50 m 위치에서 시작하여 60, 65 m에서 목표 위치에 도달하려고 하는 것으로 비교한다. 속도는 시속 10km로 고정된다. 실험에서는 400 타임스텝의 5000 에피소드를 학습한다. Fig. 8에서 결과를 보면 큰 변위를 사용하여 무게 중심을 조정하는 에이전트가 작은 변위를 사용하는 에이전트의 성능을 능가할 수 있음을 보여준다. 변위를 사용하지 않고 핸들만으로 자전거 방향을 돌릴 경우 자전거가 쉽게 쓰러진다. Fig. 9-11은 학습 과정에서 자전거의 뒷바퀴의 궤적을 보여준다. 이 시간 동안 자전거는 목표에 점차 도달해간다(파란색 선은 초기 궤적이고 빨간색 선은 늦은 궤적이다). 그러나 목표 위치의 반대 위치에서 시작하는 자전거는 종종 쓰러진다.

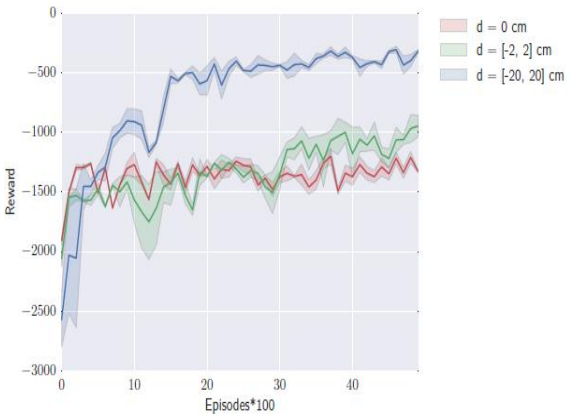

Fig. 8. Performance on difference value of displacement

자전거의 속도를 조절하려면 페달에 적용되는 힘을 알아야 한다. 다음의 성능평가 는 페달과 동시에 적용되는 힘을 학습하는 컨트롤러의 효율성을 보여준다. 자전거는 작은 변위(-2cm에서 2cm까지)만 사용하고 목표의 반대방향에서 임의로 시작된다. 평가를 위하여 속도를 학습하는 컨트롤러와 속도를 학습하지 않는 컨트롤러 간의 성능을 비교한다. 성능은 500 타임스텝의 10000 에피소드를 통하여 비교한다. 초기 속도는 초속 2.5m이다. Fig. 12는 보상의 평균을 나타낸 것이고 Fig. 13와 Fig. 14은 학습 과정에서 자전거의 궤적을 보여준다. 그림에서 볼 수 있듯이 페달 힘을 학습하면 자전거가 목표 방향의 반대 방향인 자전거를 돌리는데 도움이 된다. 자전거가 페달의 힘을 조정하여 바퀴를 돌리는 방식은 Fig. 15에 나와 있으며, 자전거가 목표 위치의 반대 방향이면 자전거가 속도를 줄이기 위하여 페달의 힘을 조정한다. 저속에서는 자전거 방향을 쉽게 돌릴 수 있다. 자전거가 목표 위치와 같은 방향으로 회전 한 후, 자전거는 가능한 빨리 목표에 도달 할 수 있는 속도를 증가시킨다. 자전거가 이미 목표 위치와 동일한 방향을 가지고 있는 경우, 자전거는 속도를 점차적으로 증가시키고 목표까지 이르게 된다.

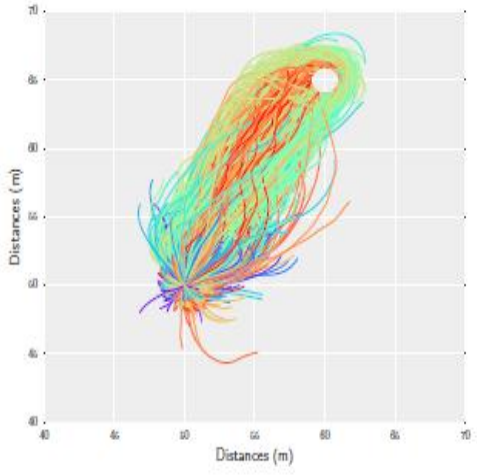

Fig. 9. d = 0.0 cm

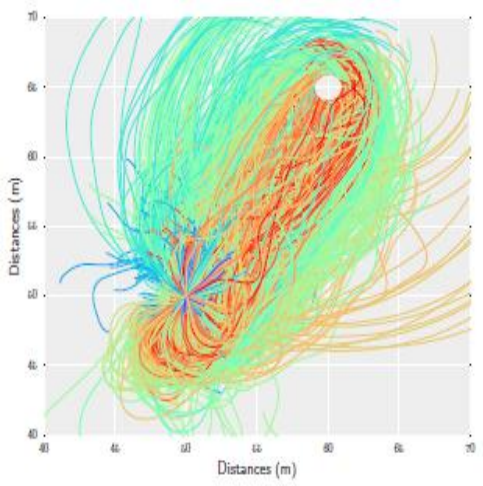

Fig. 10. d is in range −2 and 2cm

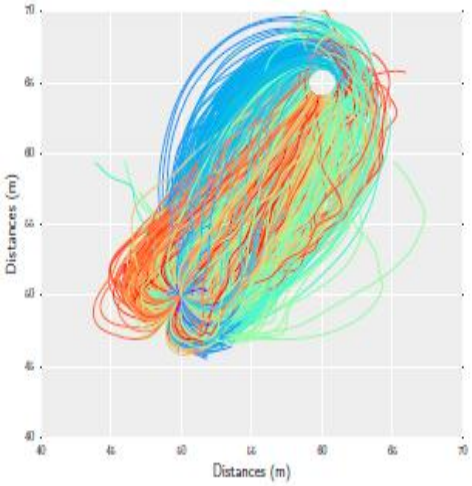

Fig. 11. d is in range −20cm and 20cm

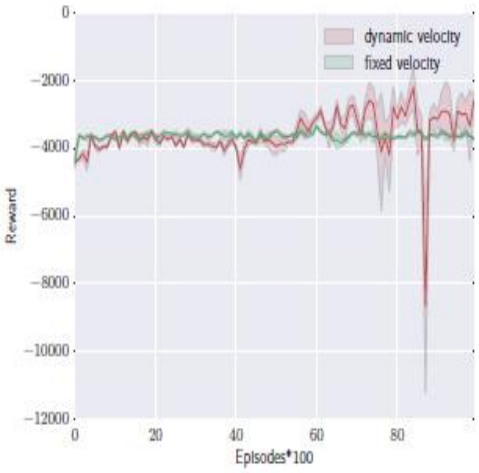

Fig. 12. Performance comparison

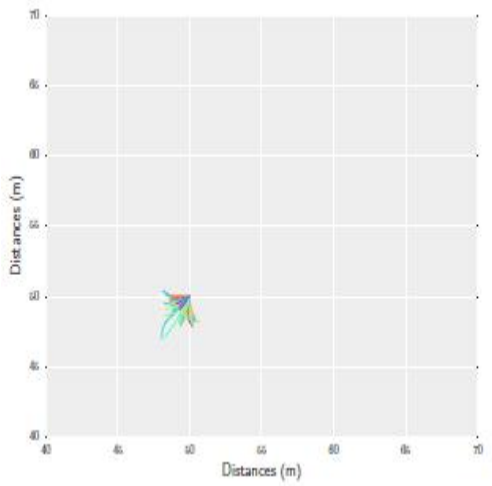

Fig. 13. Trajectories of bicycle without learning pedal's force

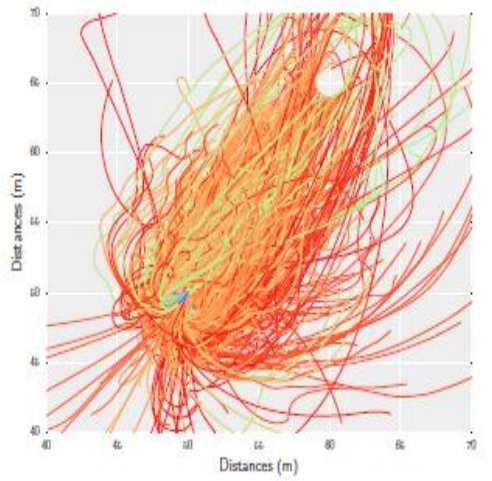

Fig. 14. Trajectories of bicycle learning pedal's force

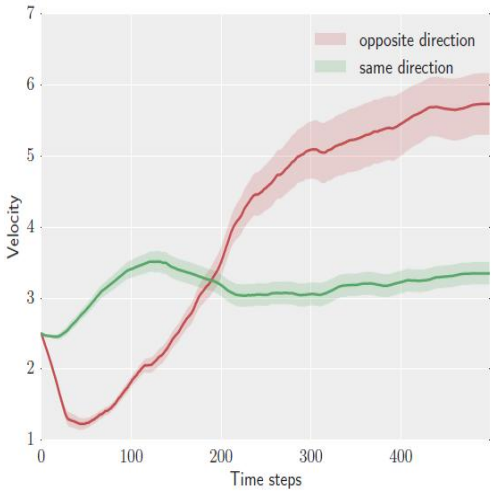

Fig. 15. The velocity of bicycle during an episode

V. Conclusions

본 논문에서는 DDPG 알고리즘을 이용하여 자전거를 제어하는 방법을 제시하였고 성공적으로 제어할 수 있음을 보였다. 심층 신경 네트워크를 채용한 컨트롤러는 자전거 핸들 바를 회전시키고, 무게 중심의 변위를 이동시키고 자전거가 쓰러지지 않고 회전 할 수 있도록 속도를 조정할 수 있다. 본 논문에서 제안한 심층 신경 네트워크 컨트롤러를 사용하는 에이전트는 어느 곳에서나 지정된 위치에 도달하고 완만하고 부드러운 궤적을 생성할 수 있었다. 실제로 사람이 자전거를 제어하는 상황과 같이 무게중심이 이동하는 경우를 고려하고 페달을 제어하는 상황을 반영하여 실험을 진행하였고 실험결과 에이전트가 무게중심을 제어하면서 이동하는 것이 가능함을 보였다. 차후에 후속 연구를 통하여 도로에서 달리는 자전거와 같은 궤적을 따라갈 수 있는 컨트롤러를 학습시키는 방법에 대하여 고려할 것이다. DDPG 알고리즘을 사용하면 올바른 범위에 속하는 스텝 크기를 선택해야하는데 만약 크기가 너무 작으면 트레이닝의 진행이 매우 느릴 것이다. 반대로 너무 크면 노이즈를 주체하지 못하여 좋지 못한 결과를 낳을 수 있다. DDPG 알고리즘은 반드시 전체 알고리즘의 성능을 향상시켜 준다고 볼 수는 없으므로 PPO를 개선하여 함께 사용한다면 좀 더 안정된 학습이 가능한 컨트롤러를 만들 수 있을 것이다. 이 문제에 대해서는 추후 연구를 진행할 예정이다.

REFERENCES

[1] L. Keo and M. Yamakita, “Controlling balancer and steering for bicycle stabilization,” 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4541-4546, Oct. 2009.

[2] J. P. Meijaard, J. M. Papadopoulos, A. Ruina, and A. L. Schwab, “Linearized dynamics equations for the balance and steer of a bicycle: a benchmark and review,” In Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, Vol 463, No. 2084, pp. 1955-1982. The Royal Society, Aug. 2007.

[3] A. Schwab, J. Meijaard, and J. Kooijman, “Some recent developments in bicycle dynamics,” In Proceedings of the 12th World Congress in Mechanism and Machine Science, pp. 1-6, 2007.

[4] J. Tan, Y. Gu, C. K. Liu, and G. Turk, “Learning bicycle stunts,” ACM Transactions on Graphics (TOG), Vol. 33, No. 4, pp. 1-16, 2014.

[5] Google Nederland, “Introducing the self-driving bicycle in the netherlands,” March, 2017.

[6] J. Randlv and P. Alstrm, “Learning to drive a bicycle using reinforcement learning and shaping,” Proceeding ICML '98 Proceedings of the Fifteenth International Conference on Machine Learning, pp. 463-471, 1998.

[7] L. P. Tuyen and T. Chung, “Controlling bicycle using deep deterministic policy gradient algorithm,” In Ubiquitous Robots and Ambient Intelligence (URAI), 2017 14th International Conference on, pp. 413-417. IEEE, 2017.

[8] J. Peters and S. Schaal, “Reinforcement learning of motor skills with policy gradients,” Neural networks, Vol 21, No. 4, pp. 682-697, May 2008.

[9] 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,” arXiv preprint arXiv:1509.02971, 2015.

[10] R. S. Sutton and A. G. Barto, “Reinforcement learning: An introduction,” Vol. 1, MIT press Cambridge, 1998.

[11] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” In ICML, June 2014.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al, “Human-level control through deep reinforcement learning,” Nature, Vol. 518, pp. 529-533, Feb. 2015.

[13] C.-L. Hwang, H.-M.Wu, and C.-L. Shih, “Fuzzy sliding-mode underactuated control for autonomous dynamic balance of an electrical bicycle,” IEEE transactions on control systems technology, Vol 17, No. 3, pp. 658-670, May 2009.

[14] G. E. Uhlenbeck and L. S. Ornstein, “On the theory of the brownian motion,” Physical review, Vol. 36, No. 5, pp. 823-841, Sep. 1930.

[15] D. P. Kingma and J. Ba. Adam, “A method for stochastic optimization,” arXiv preprint arXiv:1412.6980, 2014.

[16] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, “Continuous deep q-learning with model-based acceleration,” In International Conference on Machine Learning, pp. 2829-2838, June 2016.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” arXiv preprint arXiv:1707.06347, 2017.

[18] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” In International Conference on Machine Learning, pp. 1889-1897, 2015.

[19] M. Lu and X. Li, "Deep reinforcement learning policy in Hex game system," 2018 Chinese Control And Decision Conference (CCDC), pp. 6623-6626, 2018.

[20] E. Bejar and A. Moran, "Deep reinforcement learning based neuro-control for a two-dimensional magnetic

positioning system," 2018 4th International Conference on Control, Automation and Robotics (ICCAR), pp. 268-273, 2018.

[21] T. Yasuda and K. Ohkura, "Collective Behavior Acquisition of Real Robotic Swarms Using Deep Reinforcement Learning," 2018 Second IEEE International Conference on Robotic Computing (IRC), pp. 179-180, 2018.

Authors

Seung-Yoon Choi received the B.S. degree in Information and Communication from Korea Nazarene University, Republic of Korea, in 2010, and the M.S. degree in Computer Engineering from Kyung Hee University, Republic of Korea, in 2012, respectively. He is now working toward a Ph.D. degree at Artificial Intelligence Laboratory, Department of Computer Engineering, Kyung Hee University, Republic of Korea. His current research interests include Reinforcement Learning, Machine Learning, Optimization, and Robotics.

Tuyen Pham Le received the B.S degree in Computer Science from HCMC University of Technology, Ho Chi Minh City, Vietnam, in 2013, respectively. He is now working toward a M.S.-Ph.D. combined degree at the Artificial Intelligence Laboratory, Department of Computer Science and Engineering, Kyung Hee University, Republic of Korea. His current research interests include Machine Learning, Reinforcement Learning, and Robotics.

Tae-Choong Chung received the B.S. degree in Electronic Engineering from Seoul National University, Republic of Korea, in 1980, and the M.S. and Ph.D. degrees in Computer Science from KAIST, Republic of Korea, in 1982 and 1987, respectively. Dr. Chung joined the faculty of the Department of Computer Science at Kyung Hee University, Korea, in 1988. He is currently a Professor in the Department of Computer Engineering, Kyung Hee University. His research interests include Machine Learning, Meta Search, and Robotics.