Journal of Korea Multimedia Society Vol. 21, No. 8, August 2018(pp. 969-981)
https://doi.org/10.9717/kmms.2018.21.8.969

# 경로 탐색 기법과 강화학습을 사용한 주먹 지르기동작 생성 기법

박현준†, 최위동††, 장승호†††, 홍정모††††

# Punching Motion Generation using Reinforcement Learning and Trajectory Search Method

Hyun-Jun Park†, WeDong Choi††, Seung-Ho Jang†††, Jeong-Mo Hong††††

## ABSTRACT

Recent advances in machine learning approaches such as deep neural network and reinforcement learning offer significant performance improvements in generating detailed and varied motions in physically simulated virtual environments. The optimization methods are highly attractive because it allows for less understanding of underlying physics or mechanisms even for high-dimensional subtle control problems. In this paper, we propose an efficient learning method for stochastic policy represented as deep neural networks so that agent can generate various energetic motions adaptively to the changes of tasks and states without losing interactivity and robustness. This strategy could be realized by our novel trajectory search method motivated by the trust region policy optimization method. Our value-based trajectory smoothing technique finds stably learnable trajectories without consulting neural network responses directly. This policy is set as a trust region of the artificial neural network, so that it can learn the desired motion quickly.

**Key words:** Punching Motion, Reinforcement Learning, Physics Simulation, Character Animation, Character Control

## 1. 서 론

강화학습(Reinforcement Learning)은 환경과 에이전트가 상호작용하면서 시도와 실패를 통해 보상을 학습한다. 심층 인공신경망이 도입되기 전 강화학습은 지속적인 발전에도 불구하고 계산 복잡도, 차원의 저주등의 문제를 겪고 있었다[1]. 이러한 문제를 딥러닝(Deep Learning: DL)의 특성인 경험 재생(experience replay)을 통한 함수 근사(function approximation)를 통해 해결한 연구들이 발표되었다[2, 3]. DL과 강화학습이 결합되어 딥강화학습(Deep Reinforcement Learning :DRL) 이라고 부른다. DRL은 고차원의 물리 기반 캐릭터 애니메이션 환경에서 에이전트가 물리 기반 제어의 행동을 학습하고 강건하게 재생할 수 있는 가능성을 열어 주었다[4].

물리 기반 캐릭터 애니메이션은 물리 시뮬레이션을 통해 캐릭터와 환경을 물리적으로 현실적인 동작을 생성한다. 또한 환경의 변화와 외부의 입력으로부

※ Corresponding Author : Jeong-Mo Hong, Address: New Engineering building 10110, Dongguk Univ., 30, Pildong-ro 1-gil, Jung-gu, Seoul, Korea, TEL : +82-10-4776-2997, FAX : +82-10-4776-2997, E-mail : jmhong@dongguk. edu
Receipt date : Jun. 19, 2018, Approval date : Jul. 3, 2018

† Computer Engineering Major, Dongguk University (E-mail : hyunjun529@dongguk.edu)
†† Computer Engineering Major, Dongguk University (E-mail : wdchoi@dongguk.edu)
††† Computer Engineering Major, Dongguk University (E-mail : shjang@dongguk.edu)
†††† Computer Engineering Major, Dongguk University

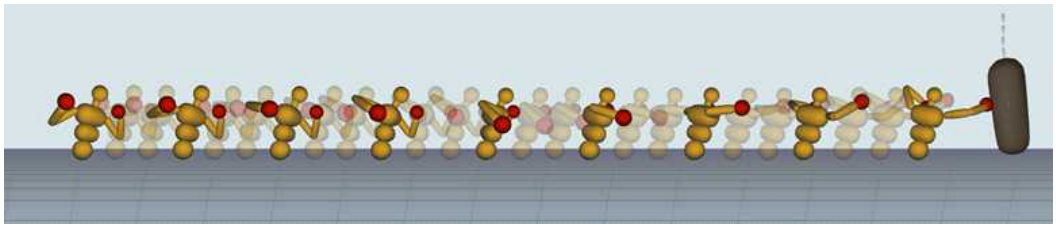


Fig. 1. Learning to punch animation for physics-based character control.

터 캐릭터의 움직임을 유지하고 민첩하게 반응할 수 있도록 캐릭터를 제어할 수 있다[5]. 하지만 인간과 같은 캐릭터의 경우에는 많은 관절을 갖고 있기 때문에 고차원 운동 방정식으로 풀어야 하므로 실시간으로 움직임을 제어하기가 어렵다. 이를 해결하기 위한 방법 중의 하나가 DRL을 이용해 환경과 상태에 따라 행동을 결정하는 정책을 구하는 것이다. DRL을 통해 학습된 정책의 장점은 상태만으로 행동을 결정할 수 있고 환경의 변화에도 민감하게 반응할 수 있는 강건함을 갖는다는 것이다. 하지만 이를 찾기까지 상당한 시간이 필요하고 국소 최저치(local minima)를 구할 가능성도 높다. 이러한 문제를 보완하기 위해 Trust region policy optimization(TRPO)[6]나 Proximal policy optimization(PPO)[7]가 등장하게 되었다. TRPO와 PPO는 이전에 찾은 최적의 경우를 신뢰 구간으로 설정하고 이를 기반으로 탐색을 수행하는 방법을 통해 강화학습에서 학습 시간과 결과의 오류를 줄일 수 있다.

우리의 목표는 DRL을 기반으로 물리 기반 캐릭터의 다양한 동작을 생성하면서 이에 소요되는 학습 시간을 단축하는 것이다. 본 논문은 물리 기반 환경에서 인간 상체를 본 뜬 캐릭터와 샌드백을 상호작용하여 Fig. 1과 같은 동작을 다양하게 생성했다. Fig. 2는 우리가 수행한 기법에 대한 개요다. 먼저 사용자가 어떤 동작을 만들 것인지 계획하여 물리 환경에 대한 보상으로 설정한다. 이후 경로 탐색 기법을 통

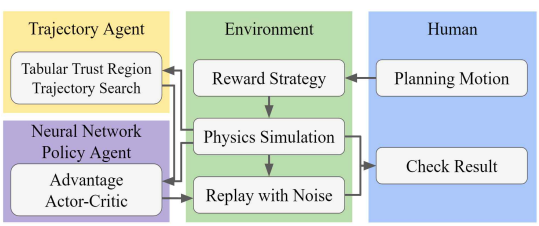



Fig. 2. Overview our pipeline.

해 한 에피소드에서 최대 보상을 얻는 행동 집합을 빠르게 구할 수 있다. 또한 이 때 구한 행동 집합으로 행동과 상태, 보상을 갖는 테이블 형태의 자료 집합(data set)으로 저장한다. 그리고 DRL 기법들 중의 하나인 Advantage Actor-Critic(A2C)에 저장된 자료 집합을 지도학습으로 초기화한 후 강화학습을 수행한다. 학습이 끝난 에이전트는 재생할 때 임의의 Noise를 행동에 주입하여 다양한 동작을 만들 수 있었다.

강화학습은 에이전트(Agent)와 환경의 상호작용이 필수적이지만 물리 기반 환경에서 환경과 상호작용하는 것은 물리 환경의 연산 때문에 부하가 높은 작업이다. 또한 DRL은 학습 초기에 좋은 정책을 찾고 안정화 하는데 많은 시간과 연산을 필요로 한다. 우리는 DRL의 학습 초기에 좋은 경험을 제공하고 물리 엔진의 연산을 최소화하기 위하여 자료 집합으로 만드는 작업을 수행했다. 이를 통해 환경에 설정된 보상이 어떤 동작을 만들지 빠르게 예측할 수 있었으며 에이전트에 좋은 초기 경험을 학습시킬 수 있었고 물리 엔진의 연산을 줄여서 환경과 상호작용의 효율성을 높일 수 있었다.

본 논문은 강화학습의 한 에피소드에서 최대 보상을 얻는 행동 집합을 빠르게 구할 수 있는 경로 최적화 기법을 제안한다. 물리 기반 환경을 사용하는 연속 행동 공간의 A2C 알고리즘에서 물리 연산을 최소화 할 수 있도록 자료 집합을 통해 초기화한 후 학습하는 기법을 제안한다. 학습된 모델의 행동에 노이즈를 일정량 주입하는 것에 비례하여 다양한 주먹 지르기 동작을 생성할 수 있음을 실험 결과와 함께 제시한다. 2장에서 우리 연구와 관련된 물리 기반 캐릭터 애니메이션과 강화학습의 기술들을 소개하고 3장에서는 경로 탐색에 들어가는 Tabular Trust Region과 이를 보완하기 위한 Trajectory Smoothing 방법을 설명한다. 4장에서는 본 논문에서 사용한 인공신경

망의 모델과 경로 지도학습, A2C를 설명한다. 5장에서는 실험 결과를 보이고 6장에서 결론과 향후연구로 마무리를 하겠다.

## 2. 관련 연구

물리 기반 캐릭터 애니메이션 분야는 물리적으로 타당한 동작과 움직임을 생성하는 연구를 한다. 인간을 본 뜬 가상의 캐릭터가 물리 시뮬레이션 환경에서 무게 중심을 잃지 않고 걸을 수 있도록 제어하는 연구가 이어져 왔다[8-12]. 이와 함께 캐릭터의 상반신을 제어하여 다양한 동작을 만드는 것도 중요하게 연구되어 왔다. [13]는 모션 데이터를 기반으로 환경과 상호작용할 수 있는 복싱 애니메이션을 만들었다. Hwang et al. [14]는 역진자 모델을 사용하여 두 캐릭터의 격투 애니메이션을 만들었다.

강화학습은 에이전트(agent)가 환경(environment)과 상호작용하면서 현재 상태(state)를 바탕으로 보상(reward)을 최대화할 수 있는 행동을 선택하는 방법이다. 매 단계에서 에이전트가 행동 $a$를 수행하면 환경은 보상 $r$과 다음 상태 를 $s$ 반환한다. 에이전트는 이를 반복하여 한 에피소드(episode)에서 얻을 수 있는 보상을 최대로 하는 것을 목표로 한다. 정책(policy) $\pi$은 주어진 상태 $s$에서 어떤 행동 $a$을 선택하는가 결정하는 $a=\pi(s)$ 꼴의 함수이다[15]. DL 이전의 강화학습은 상태의 크기를 줄이는 방식으로 학습 시간을 줄이는 방법이 연구되었다[16]. DRL은 심층 인공신경망을 정책 함수 근사에 사용한다. 이를 통해 DQN은 이산 행동 공간(discrete action space)에서 경험 재생을 효율적으로 역전파함으로써 사람 수준의 게임 조작이 가능해졌다[2].

강화학습에서 연속 행동 공간은 행동을 $-1 \leq a \leq 1$와 같이 실수 범위로 선택하는 것이다[17]. 이산 행동 공간(deterministic action space)을 다루는 강화학습이 마르코프 결정 프로세스를 풀기 위해 유한한 행동 집합을 고려하는 것과 차이가 있다. 이 문제를 해결하기 위해서 일반적으로 Policy Gradient와 Actor-Critic 기법을 사용한다. 연속 행동 공간을 풀기 위한 Policy Gradient 기법[18-20]은 학습하고자하는 모델을 몰라도 정책의 파라미터를 학습할 수 있는 model-free 특성을 이용한다. 그리고 Actor-Critic 기법은 실제 행동하는 행동 정책(Actor)과 보상을 평가하는 가치 함수(Critic)로 나누어서 학습을 하는 방법이다.

A3C(Asynchronous Advantage Actor-Critic)[21]는 행위자(Actor)를 비동기적으로 더 많이 수행하여 경험의 표본을 많이 쌓아 학습 속도를 높였다. DPG[22]는 연속 행동 공간의 강화학습에서 정책 학습 확률적 정책(stochastic policy) 대신 결정론적 정책(deterministic policy)를 사용해 학습에 필요한 표본의 수를 줄여 성능을 높였다. 그리고 DDPG[23]는 DPG에 DQN의 심층 인공신경망을 도입하여 가치 함수 근사에 사용했다. TRPO[6]와 PPO[7]은 정책의 성능이 낮게 나오는 구간의 표본 데이터를 최적화했다. 이를 위하여 정책 함수를 학습할 때 loss function에 대체 목적 함수(surrogate objective function)를 활용했다. PPO는 Nicolas et al. [24]에서 다양한 환경과 보상을 통해 캐릭터의 locomotion들을 만들 수 있음을 보였다.

물리 기반 캐릭터 애니메이션 분야는 물리 시뮬레이션을 수행하는 물리 환경과 물리 기반 제어를 수행하는 에이전트를 통해 강화학습을 적용한 연구들이 이뤄지고 있다. Tan et al. [25]는 인간을 본 뜬 캐릭터가 자전거를 타는 물리 제어 기법을 제안했다. Wang et al. [26]은 영상처리로 입력된 실제 사람의 동작과 가상 캐릭터의 칼싸움을 상호작용했다. Liu et al. [27]는 물리 기반 환경에서 다양한 물체 위에 선 캐릭터가 무게 중심을 잡는 여러 동작을 생성했다. Won et al. [28]는 공기역학과 물리 기반 모터 제어를 통해 가상 생물의 날갯짓을 학습 시켰다. Xue et al. [29]는 인간 하체를 본뜬 캐릭터의 이동 동작(locomotion)을 제어하기 위한 계층적 학습 기반 프레임워크를 제안했고 축구를 하는 동작까지 성공했다.

## 3. Trajectory Search

본 논문은 물리 시뮬레이션을 사용하는 강화학습 환경의 한 에피소드에서 최대 보상의 행동 집합을 구하는 TTR(Tabular Trust Region) 기법을 제안한다. 상태에 대한 정책을 사용하지 않고 에피소드에서 각 단계(step)의 행동 값을 저장함으로써 DRL에 비해 적은 연산 회수로 행동 집합을 구할 수 있다. 이는 빠르게 최적의 행동 집합을 찾을 수 있지만 DRL처럼

다양한 동작을 만들 수 없다. 반면에 DRL은 하나의 유의미한 동작으로 수렴하기까지 오랜 시간과 물리 시뮬레이션 연산을 필요로 한다. 이 경우 DRL로 찾은 결과가 의도한 바와 다를 경우 처음부터 다시 환경과 보상을 설계하고 실험을 반복해야하기 때문에 비용이 많이 들어가게 된다. 따라서 우리는 TTR를 통해 환경과 에이전트를 빠르게 검증한다. 그리고 최대 보상을 얻을 수 있는 경로에 대한 자료 집합은 동일한 조건의 DRL에서 초기 학습의 자료 집합으로 사용할 수 있다.

Fig. 3은 Trajectory Search의 동작 방법을 시각화했다. 좌측은 처음 경로를 찾을 때 넓은 범위로 경로를 탐색한 것이다. 첫 시도에선 어느 경로가 좋은가 알 수 없으므로 탐색 범위를 결정하는 파라미터 σ를 크게 설정한다. 우측은 더 좋은 보상을 발견한 경우 해당 경로를 최적으로 가정하고 그 경로를 중심으로 탐색할 수 있도록 σ를 줄여 탐색 범위를 좁힌다.

본 논문에서 경로(Trajectory)는 물리 환경에서 관절 제약(Constraints)에 따른 끝점의 위치 값이다. 시공간 최적화(spacetime optimization)이라고도 불리는 이 문제는 불연속적인 이벤트 처리를 최적화한다. 여기서 시공간(spacetime)은 불리는 불연속적 이벤트는 프레임과 프레임 사이의 간격을 의미하며 강화학습에서 행동 값을 적용하기 위한 각 단계를 나누는 단위다. TTR에서 행동은 각 단계마다 정하지만 보상은 한 에피소드에서 총합으로 평가한다.

### 3.1 Tabular Trust Region

Tabular Trust Region(TTR)은 강화학습의 환경, 에이전트 구조를 따르며 행동을 탐색할 때 정규분포 $N$를 사용한다. 그리고 최대 보상을 얻는 행동과 보상, 상태를 단계별로 테이블 형태의 자료 집합으로 저장한다. DRL과 가장 큰 차이점은 행동과 보상에 대해서 정책 함수를 사용하는 대신 직접 행동을 저장하고 에피소드를 반복할 때마다 σ를 다시 계산하는 부분이다.

경로 $T$는 한 에피소드에서 에이전트가 수행하는 행동 $a$와 행동에 대한 보상 $r$을 프레임 재생 순서에 따른 각 단계 저장한다. TTR에서 경로를 구할 때 상태 $s$는 사용하지 않지만 DRL 등에 필요한 경우 경로를 재생하여 저장한다. 경로 탐색은 정규분포 $N$에서 평균 μ과 표준편차 σ를 파라미터로 사용한다. μ의 범위는 -1~1이며 0으로 초기화하며 σ는 1.0로 초기화한다. μ는 직접 행동 값으로 사용되며 σ는 탐색 범위로 사용한다. 더 나은 경로를 발견할 경우 μ는 직접 대입하여 바꾸고 σ는 같은 반복 단위의 탐색 경로들과 비교한 기대값을 바탕으로 갱신한다.

Algorithm 1은 TTR을 통해 경로 탐색하는 기법을 설명한다. $n_m$는 한 반복에서 탐색하는 경로의 수이며 이는 같은 조건에서 탐색하므로 병렬로 처리할 수 있다(line 4). 각 경로는 탐색할 행동을 단계별로 저장된 μ와 σ 파라미터의 정규분포를 통해 구한 후 Trajectory Smoothing을 적용하여 최종 결정한다(line 5). 탐색하기로 결정한 행동을 환경에 상호작용하여 수행한 후 이에 대한 $r$을 얻어 저장한다(line 6). 지금까지 최대 보상을 얻은 경로보다 더 큰 보상을 얻은 경로를 발견하면 최대 보상 경로를 갱신한다(line 9).

한 반복에서 경로 탐색이 끝난 후 최대 보상 경로의 μ는 최대 경로의 μ를 직접 대입하여 저장한다(line

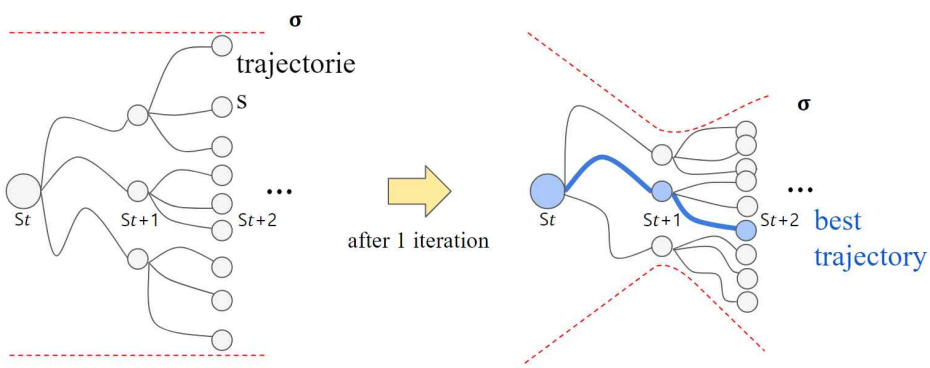



Fig. 3. Example of Trajectory Search.

**Algorithm 1.** Tabular Trust Region Trajectory Search

$n_t$ : number of Trajectory Searches
$n_e$ : number of episode's step
$T_{best}$ : best trajectory in search result

1: **procedure** TRAJECTORY SEARCH
2: **while** not terminate **do**
3: **for** $i=1\cdots n_t$ **do**
4: **for** $j=1\cdots n_e$ **do**
5: $T_j, a_{ij} \leftarrow$ TrajectorySmoothing($N(\mu_{bj},\sigma_{bj})$)
6: $r_{ij} \leftarrow$ Environment($a_{ij}$)
7: **end for**
8: **if** $T_{best}(\text{sum}(r_{best})) < T_i(\text{sum}(r_i))$ **then**
9: $T_{best} = T_i$
10: **end if**
11: **end for**
12: $\mu_{best} \leftarrow T_{best}$
13: **for** $i=1\cdots n_t$ **do**
14: $\hat{A} = T_i(\text{sum}(r_i)) - T_{best}(\text{sum}(r_{best}))$
15: **for** $j=1\cdots n_e$ **do**
16: $\sigma_{bj} \leftarrow$ maximizeExpectation($T_j, T_{best}, \hat{A}$)
17: **end for**
18: **end for**
19: **end while**
20: **end procedure**

12). 이번 반복에서 탐색한 경로들과 현재 최대 보상 경로의 보상 오차를 구하여 Advantage $\hat{A}$를 계산한다(line 14). σ는 $\hat{A}$를 통해 다음 탐색 범위를 결정하기 위한 기대값 범위를 갱신한다(line 16). 이 때 기대값 계산에는 PPO 기법의 클램핑 조건을 사용한다. 앞서 구한 $\hat{A}$는 σ의 급격한 변화를 막기 위해서 사용한다. σ가 기본 값에 비해서 너무 크게 변할 경우 $\hat{A}$와 비례해 일정량 이상 바뀌지 않도록 클램핑한다.

### 3.2 Trajectory Smoothing

경로 Trajectory Smoothing은 에피소드의 각 단계에서 생성되는 행동들을 스무딩하는 최적화 함수이다. 경로내 평균이 크게 형성될 경우 보상은 유사하지만 대체로 사람 눈에 부자연스러운 동작이 나온다. 물리 시뮬레이션 환경에서 인간과 유사한 질량, 길이 등을 적용한 캐릭터지만 실제 세계만큼의 복잡도를 반영하지 못하기 때문이다. 이를 방지하기 위해 탐색한 행동을 지금까지 누적된 행동의 평균에 가깝게 조정하여 부드럽게 선택되도록 최적화 함수를 적용한다. 현재 단계 $t$에서 $n$만큼 이전 행동을 참조하며 $w$는 $t$에서 c만큼 전의 특정 단계를 의미한다 (Equation. 1).

$$a_t = \frac{\sum_{w_i=(t-n)}^{t} w_i a_{t-w_i}}{\sum_{i=1}^{n} i} \quad (1)$$

TTR를 통하여 얻은 모션과 실제 주먹 지르기 모션의 유사성을 확인하기 위하여 실제 모션 캡처 데이터와 비교하였다. 비교에 사용한 모션 캡처 데이터는 Carnegie-Mellon Mocap Database(CMU)에서 얻었다. 해당 데이터는 20대 남성에 대해서 모션 캡처를 진행했다. Table 1은 실제 모션을 캡처한 남성의 신체 정보와 논문에서 사용한 캐릭터의 정보를 보여준다. Armstrong et al. [29]을 참고하여 우리가 사용한 캐릭터와 비슷한 신체 조건의 남성 모션 캡처 데이터를 사용했다.

Fig. 4의 3차원 그래프에서 파란 선은 모션 캡처한 주먹의 궤적이고 빨간 선은 TTR를 사용하여 얻은 모션이다. 두 개의 궤적은 모두 처음에 주먹을 뒤로 당겨서 에너지를 축적하고 다음에 팔을 펴서 주먹을 측면으로부터 반원형으로 돌려 목표물을 타격한다. 궤적을 비교하면 TTR를 사용하여 얻은 모션이 실제의 훅 모션과 유사하고 보기에 자연스럽게 생성된다.

Table 1. Compare of our character and real human

| | mass(kg) | height(m) | arm length(m) |
|---|---|---|---|
| Person | 75 | 1.71 | 0.79 |
| Character | 80 | 1.74 | 0.84 |

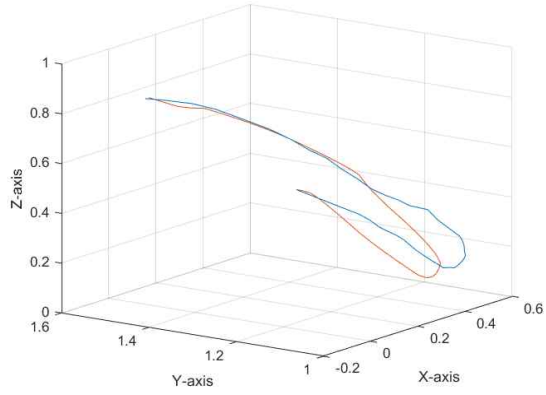



Fig. 4. Compare our trajectory with MoCap data.

## 4. 인공신경망을 활용한 학습

TTR은 최적의 보상을 얻는 경로를 빠른 시간에 구할 수 있지만 비슷한 보상을 얻는 다양한 경로를 실시간으로 생성할 수 없다. 반면 신경망는 정책이 수렴한 이후 행동에 노이즈를 주입해도 강건하게 동작하며 실시간으로 재생시킬 수 있다. 본 논문에서는 물리 기반 케릭터 애니메이션에 최적화된 경로를 구하고 실시간으로 동작을 재생하기 위해서 신경망에 지도 학습시킨 후 A2C 기법을 적용한다. 학습된 신경망의 정책은 TTR를 통해 생성된 경로와 유사한 동작을 보이도록 수렴하며 재생할 때 정책에 σ를 노이즈로 섞어 다양한 동작을 생성한다. 우리는 Fig. 5와 같이 신경망 모델을 구성하고 완전연결신경망를 사용한다. 이렇게 μ와 σ를 출력을 구분함으로서 재생시 노이즈를 σ에 설정할 수 있다.

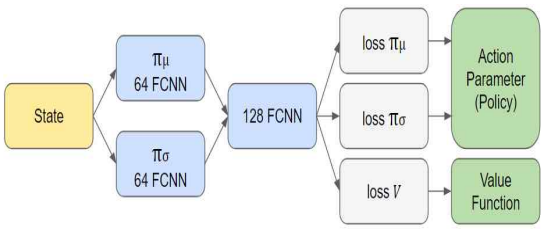



Fig. 5. our Neural Network Model.

### 4.1 경로를 활용한 신경망 정책 학습

강화학습은 보상을 최대화할 수 있는 정책을 찾기 위해 임의의 행동을 취하거나 기존 경험을 바탕으로 더 나은 행동을 탐색한다. 물리 시뮬레이션을 강화학습의 환경으로 사용할 경우 에이전트가 탐색한 후 보상과 상태를 받기 위하여 반드시 물리 시뮬레이션 연산을 수행해야한다. 이로 인하여 물리 시뮬레이션 환경을 사용하는 강화학습은 에이전트가 환경과 상호작용할 때 부하와 병목이 발생한다.

초기 신뢰 구간 형성에 드는 시간을 최소화하며 물리 환경 연산량을 줄이기 위하여 TTR에서 얻은 경로를 자료 집합으로 지도학습을 수행한다. TTR에서 얻은 경로를 각 단계의 상태, 보상, 행동으로 이루어진 테이블 형태의 자료 집합을 A2C 기법에 적용한다. 최적의 경로를 자료 집합으로 사용하면 물리 환경을 직접 연산하지 않으면서 이미 검증된 좋은 경험을 사용하므로 빠른 초기 학습이 가능하다.

우리는 A2C 기법에서 가치 함수를 갱신할 때 TTR에서 σ를 최적화 했던 것과 동일한 PPO 기법을 사용한다. 이를 통해서 우리는 TTR에서 형성된 결과와 이 자료 집합을 통해 학습된 정책이 유사성을 가질 수 있도록 한다. 정책은 TTR 탐색과 마찬가지로 정규분포 $N$을 사용하며 μ와 σ를 파라미터로 가진다. 하나의 경로가 아니라 복수의 경로를 사용해 초기화를 수행할 수도 있지만 이는 추후 과제로 남긴다.

### 4.2 신경망 정책의 강화학습과 재생

우리는 최적 경로를 생성하여 이를 자료 집합으로 만들고 지도학습을 수행했다. 지금까지 신경망을 활용한 강화학습 알고리즘의 결과를 미리 예측하고 물리 시뮬레이션 연산을 최소화하기 위한 방법들을 제안했다.

최적 경로만 지도학습한 에이전트는 스스로 환경에 대한 경험을 획득하지 못한 상태이다. 새로운 경험을 에이전트가 스스로 행동을 선택하도록 하여 다시 학습시켜야 노이즈가 주입되어도 유효한 보상을 얻는 강건한 에이전트가 생성된다.

먼저 학습을 수행하는 과정은 Algorithm 2와 같다. 먼저 경로를 학습할 때는 행동 $a$, 상태 $s$, 보상 $r$에 대한 정보를 환경이 아니라 최적 경로의 자료 집합에서 얻는다(line 4). 상태 $s$는 물리 환경을 통해 에이전트의 행동에 대한 상태를 저장하고 있다. 우리는 상태로 주먹의 위치(position)나 속도(velocity), 충격량(impulse) 그리고 샌드백의 위치를 사용했다.

미리 구한 자료 집합(DataSet)을 충분히 학습이 된 후에는 A2C 알고리즘을 수행한다. 지도학습이 끝난 에이전트는 A2C 알고리즘에 따라서 행동을 직접 선택한다((line 7). 신경망을 직접 강화학습 할 때 $\hat{A}$는 지금까지 얻은 최고 보상과 이번 에피소드의 보상의 차로 사용했다(line 15). 그리고 최대 기대 값을 평가하는 가치 함수 $V$를 같이 갱신한 후 행위자의 행동 μ를 갱신한다(line 17).

Algorithm 3은 학습이 끝난 에이전트를 재생하는 기법을 설명한다. 본 논문은 학습이 끝난 에이전트가 동작을 다양하게 생성하기 위해 학습이 끝난 에이전트에 σ를 조절하는데 이를 노이즈로 정의한다. 우리는 재생시 학습이 끝난 에이전트를 불러온 후(line 2) 임의로 노이즈 σ를 설정하고(line 5) 정책인 정규

**Algorithm 2.** Learning A2C Agent

```
procedure LEARNING A2C WITH TRAJECTORY
  while not terminate do
    if learn from Trajectory then
      a,s,r ← Dataset obtained through TTR, T
    else
      for i = 1 ··· n_e do
        s_i = Environment(a_{i-1})
        μ_i,σ_i ← π(s_i)
        a_i = N(μ_i,σ_i)
        r_i ← Environment(a_i)
      end for
    end if
    if r_best < sum(r) then
      r_best = sum(r)
      a_best = a
    end if
    Â = r_best − sum(r)
    for i = 1 ··· n_e do
      V,μ_i,σ_i ← maximizeExpectation(a,a_best,Â)
    end for
  end while
end procedure
```

**Algorithm 3.** Replay A2C Agent

```
procedure REPLAY A2C
    N ← load learned A2C Agent
    for i = 1 ··· n_frame do
      μ_i ← π(s_i)
      σ ← set Noise
      a_i = N(μ_i,σ)
      r_i ← Environment(a_i)
    end for
end procedure
```

분포에서 평균 μ의 분산으로 행동을 구한다. 출력된 행동에서 μ는 최적의 행동이며 σ는 표준편차만큼 이 행동을 조금씩 어긋나게 만든다. 하지만 μ는 다음 단계에서도 최적의 행동을 계속 선택해 유효한 보상을 얻을 수 있다.

## 5. 실험 결과

본 연구의 실험은 Inter(R) Core(TM) i7-7820X CPU와 GeForce GTX 1080Ti GPU에서 수행됐으며 물리 환경으로 Bullet Physics 2.86을 사용하였다. 실험 환경에서 TTR은 3분 51초, A2C 알고리즘을 활용한 신경망 학습은 100 에피소드를 수행하는데 1시간 10분이 소요됐다. 우리는 약 10000 epoch 정도 지도학습한 후 A2C를 수행하여 좋은 결과를 얻을 수 있었다. Fig. 6은 TTR으로 지도학습한 후 A2C를 수행한 경우와 A2C만 수행한 경우에 대해 에피소드의 평균 보상을 측정했다. 두 조건을 같은 epoch에서 비교하기 위해 TTR을 사용한 A2C는 지도학습을 수행한 만큼 epoch를 수행한 시점에서 A2C만 사용한 경우와 비교했다. 이 실험 결과를 통해서 TTR를 사용하는 것이 A2C만 사용하는 것 보다 유용함을 확인했다. 또한 TTR로 지도학습하는 동안은 물리 시뮬레이션을 수행하지 않아 연산량과 시간의 소모를 줄일 수 있었다.

실제 사람이 움직이는 동작과 유사한 움직임을 만들기 위해 캐릭터 모델의 비율과 질량을 Armstrong et al.[30]에 기재된 평균 수치로 Fig. 7처럼 각 부분의 무게(kg)를 설정하였다. 우리는 Ragdoll 상태에 상체만 가진 3D 모델을 사용했다. 그리고 캐릭터의 골반은 고정시킨 상태로 양손을 물체를 연결하여 균형을 잡을 수 있도록 만들었다.

본 논문에서 사용한 캐릭터 모델은 인간 상체를 본 떠 타격 동작을 생성하기 위해서 10개의 관절로 이루어졌다(Fig. 7). 관절들은 회전축에 따라 4가지로 나눠서 연결하였다. 타격을 할 때 회전이 필요 없는 머리, 목, 가슴, 손목은 0 자유도인 Weld로 관절을 연결하였다. 나머지는 사람의 형태에 따라 팔꿈치는 1자유도의 회전 관절, 허리는 2자유도의 유니버셜 관절 그리고 어깨는 3자유도인 볼 소켓 관절을 이었다.

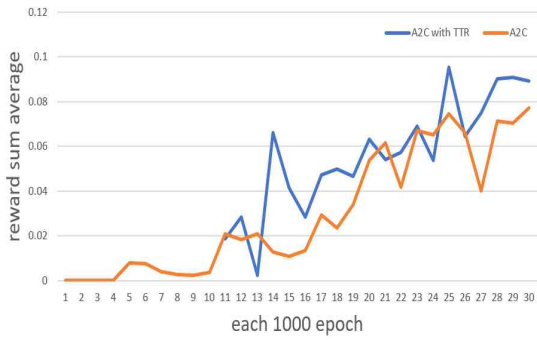



Fig. 6. Compare A2C with TTR and A2C only.

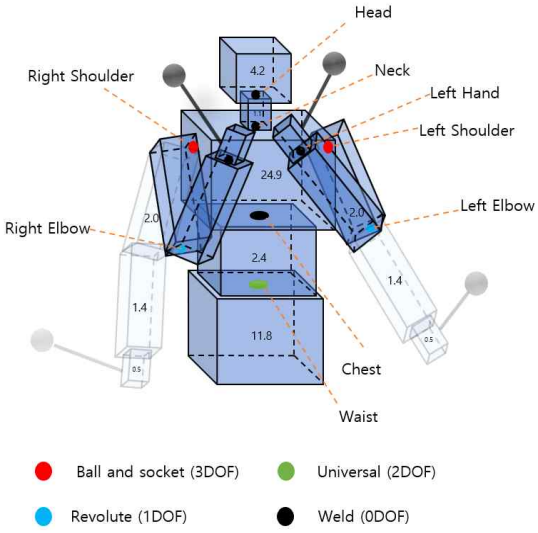



Fig. 7. The dynamic models of our upper body character.

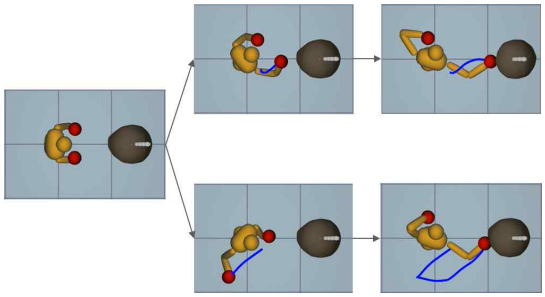


Fig. 8. Compare of reward direction.

### 5.1 Trajectory Search 환경 설계

우리는 캐릭터의 주먹의 움직임에 사용자의 의도를 반영하기 위해 샌드백과 상호작용하여 발생하는 값들을 보상으로 설정했다. 본 논문에서 보상 값을 만드는데 사용한 물리 환경의 파라미터들은 주먹과 샌드백의 무게중심 사이의 거리, 주먹이 샌드백이 충돌했을 때 발생하는 충격량, 주먹의 속도와 3차원 좌표공간의 절대 좌표를 사용했다.

보상에서 각각의 파라미터를 의도에 맞게 우선순위를 정해주는 것이 필요하다. 이 우선순위에 따라 다양한 동작이 만들어진다. Fig. 8은 보상의 우선순위에 따라서 경로가 각각 다르게 생성되는 것을 보여준다. 위의 예제는 정면의 보상이 높은 반면 아래의 예제는 측면의 보상이 높게 설정한 사례이다.

Fig. 9는 에이전트에게 보상을 Table 2와 같이 설정하여 여러 움직임을 학습한 결과이다. (a)와 (b)는 거리와 속도에 따라 보상을 주도록 만들었다. 에이전트는 표적과 거리가 가깝고 z축으로 향하는 속도가 빠르면 높은 보상을 받게 된다. 그 결과 가상의 캐릭터는 권투의 스트레이트[31]처럼 최단 거리를 이용하여 신속, 정확하게 표적을 타격하도록 학습되었다. 다만, (a)는 에피소드의 길이를 50 프레임으로 제한하여 힘을 축적하는 예비 동작없이 바로 샌드백을 타격하는 반면에 (b)는 100 프레임으로 늘려주었기 때문에 허리의 회전력도 이용하여 샌드백을 타격하것을 확인할 수 있었다. 한 에피소드에서 단계 단위는 프레임의 길이로 조절되며 에이전트는 5 프레임마다 하나의 행동을 선택하게 된다. 그러므로 에피소드가 길어질수록 행동의 범위는 넓어지고 힘을 최대한으로 높이기 위해 준비하는 동작이 길어지면서 거의 마지막 프레임에 샌드백을 타격하게 된다.

(c)는 권투에서 사용하는 훅과 유사한 움직임을 학습하였다. 권투에서의 훅은 허리와 어깨를 회전하는 힘으로 상대방을 측면에서 타격하는 위력있는 타격 기술이다[32]. 우리는 이를 목표로 에이전트에게 표적과의 거리와 x와 y축으로 향하는 속도를 보상으로 주었다. 이대로 학습한 캐릭터는 (c)처럼 먼저 오른쪽으로 주먹을 움직이고 허리와 어깨를 회전하는 힘을 사용하여 샌드백의 측면을 힘있게 타격했다. 그림(d)는 권투의 스윙과 유사한 움직임을 보여준다. 권투의 스윙은 팔을 펴서 주먹을 측면으로부터 반원형으로 돌려 옆으로 때리는 것이다. 우리는 충격량과 X와 Z축으로 향하는 속도를 보상으로 정의하였다.

Table 2. Reward of each case to generate motion

| | Frame | Reward |
|---|---|---|
| (a) Straight | 50 | $\max(0.0,(0.3-\delta_d))^2 * 10.0 v_z$ |
| (b) Straight | 100 | $\max(0.0,(0.3-\delta_d))^2 * 10.0 v_z$ |
| (c) Hock | 50 | $\max(0.0,(0.3-\delta_d))^2 * 10.0 (v_z + 0.75 v_z)$ |
| (d) Swing | 50 | $\max(0.0,(0.3-\delta_d))^2 * 500.0 (v_z + 0.5 v_z)$ |

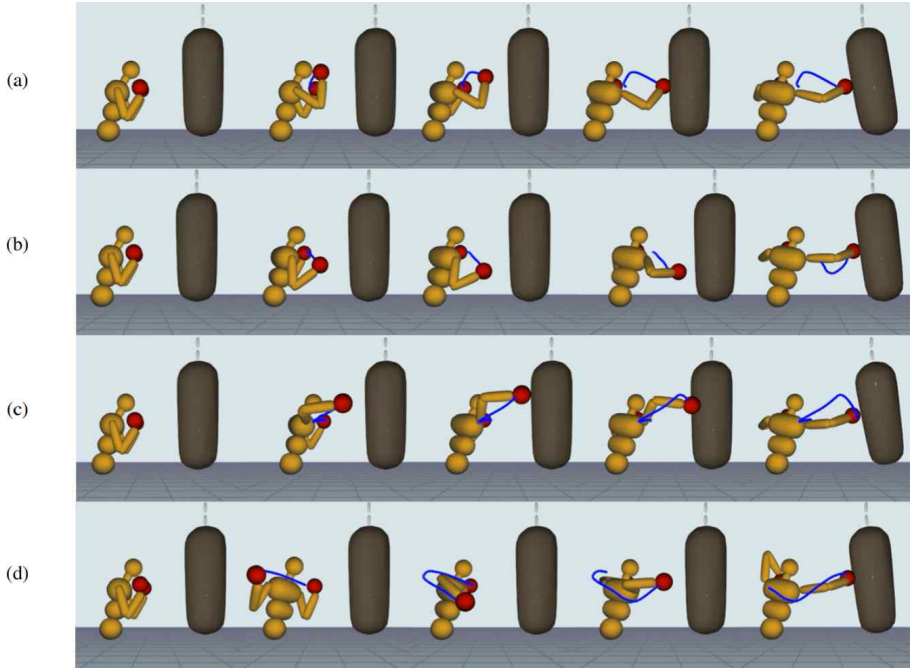



Fig. 9. Result of generated motion. (a) (b) (c) (d).

그 결과 (d)처럼 캐릭터는 주먹을 뒤로 빼서 목표물과의 거리를 최대로 만든 다음 반원형으로 허리와 어깨를 회전과 동시에 목표물의 측면을 타격했다.

### 5.2 Replay with Noise

우리는 신경망 정책으로 학습된 정책에서 다양한 동작을 생성하기 위해서 학습이 끝난 후 정규분표 정책에서 σ를 노이즈로 주입했다. 정책이 강건하게 학습된 에이전트는 노이즈를 주입한 정도에 따라서 일정량의 보상과 자연스러운 동작을 출력할 수 있었다.

Fig. 10은 노이즈를 주입하고 재생하여 생성되는 동작들의 예제와 생성되는 동작들의 영역에 대한 실험 결과이다. 우리는 노이즈가 주입되면 최선이 아닌 행동을 출력하지만 그에 맞춰 오차를 수정하면서 보상을 획득하는 것을 확인했다. 또한 노이즈의 주입 정도에 비례하여 동작이 생성되는 영역이 넓어지고 환경에서 얻는 보상 값이 낮아짐을 확인했다. 만약 노이즈가 0으로 설정하면 μ만 사용하여 같은 동작을 반복하는 것으로 정의한다. 즉 학습된 에이전트가 노이즈가 없이 출력하는 최선의 행동이 노이즈가 0일 때이다.

본 논문은 노이즈를 주입한 정도에 따라 에이전트의 행동이 일정한 동작을 수행하는가에 대한 비교 자료를 제시한다. Fig. 11는 학습된 신경망 정책에 노이즈를 0.1단위로 0.1부터 0.9까지 주입한 후 10,000회 재생한 결과이다. 좌측은 동일한 환경과 보상 조건에서 Trajectory Search를 수행하여 얻은 경로의 보상 값에 대한 노이즈를 주입한 정책의 보상 값의

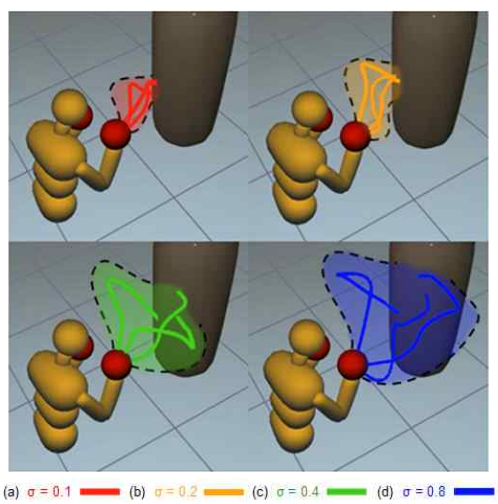



Fig. 10. Result of replay with Noise(σ ) rate.

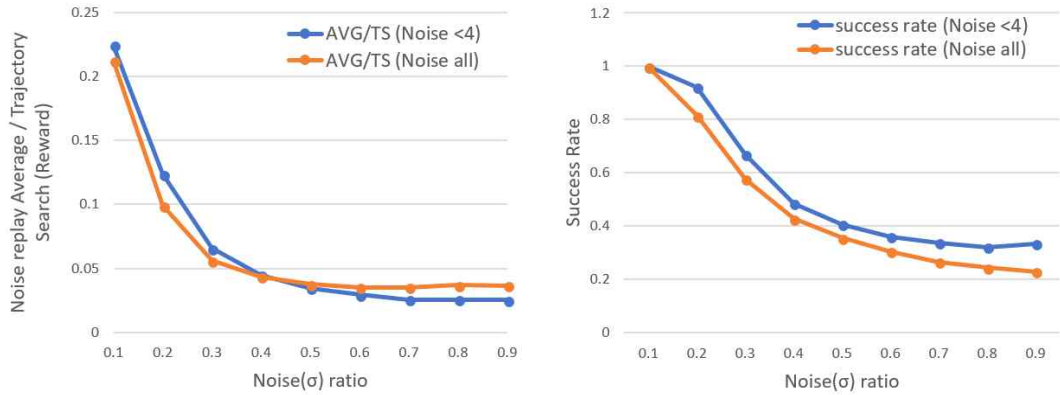



Fig. 11. Result of replay with Noise ( $\sigma$ ) rate.

비율이다. 이를 통해서 노이즈를 주입하는 비율에 따라 얻어지는 평균 보상이 상관관계를 보임을 확인할 수 있다. 우측은 에피소드의 성공률에 대한 결과다. 우리는 성공을 보상이 0이 아닌 에피소드로 정의했다. 그 후 총 재생 횟수에서 성공한 에피소드의 비율을 구하여 비교했다.

(Noise < 4)는 한 에피소드에서 총 10번의 행동 중 처음 4번 행동에만 노이즈를 섞고 이후 6번의 행동은 노이즈를 주입하지 않은 결과이다. 우리는 처음 몇 프레임에만 섞은 후 Noise 0으로 재생하는 것이 더 안정적으로 동작이 생성됨을 확인했다. 노이즈를 주입하고 재생하는 실험 결과 노이즈를 섞는 비율에 따라서 동작이 점진적으로 불안정해지며 이를 사용자가 조절할 수 있음을 확인했다.

## 6. 결 론

우리는 제시한 기법을 이용하여 캐릭터에게 샌드백을 타격하도록 학습시켰고 강화학습으로 학습된 하나의 에이전트가 물리 환경에서 다양한 주먹 지르기 동작을 생성하는 기법을 제안한다. 사용자는 보상만을 설정해주면 고차원의 물리 방정식의 지식이 없어도 캐릭터 스스로 권투와 유사한 동작을 학습되는 것을 확인했다. 그리고 사용자가 원하는 방향에 따라 보상 전략을 설정해주면 다양한 동작을 만들 수 있음을 확인했다.

물리 환경에서 캐릭터의 애니메이션을 생성하기 위한 강화학습 알고리즘의 행동, 보상, 상태를 설계했다. 이에 따라 우리는 주어진 환경에서 보상을 최대로 얻을 수 있는 최적 경로 탐색 방법으로 Tablular Trust Region을 제안했다. 그리고 이 과정에서 캐릭터가 보다 자연스러운 움직임으로 만들어주기 위한 Trajectory Smoothing 기법도 제안했다. 신경망의 정책을 학습할 때 A2C 알고리즘을 적용했고 학습 후 정책을 재생할 때 노이즈를 섞어 여러 동작을 생성할 수 있음을 실험결과와 함께 제안한다.

우리는 실험을 진행하면서 Fig. 12와 같이 실패한 경우가 발생했었다. (a)는 강체가 물리 시뮬레이션의 오류로 정상적인 강체의 위치를 벗어났다가 원래 자리로 돌아올 때 발생하는 속도를 이용하도록 경로가 학습된 경우였다. 이와 같이 물리 연산의 오류를 활용해서 높은 보상을 찾는 경우가 종종 발생했었다. 또 (b)의 경우 보상 조건이 샌드백의 무게 중심이었는데 무게 중심과 닿지 않게 샌드백을 밀어낸 후 반동을 이용해서 가격하여 충격량을 최대화하는 경로 탐색 결과였다. 이런 종류의 실패를 피하기 위해서

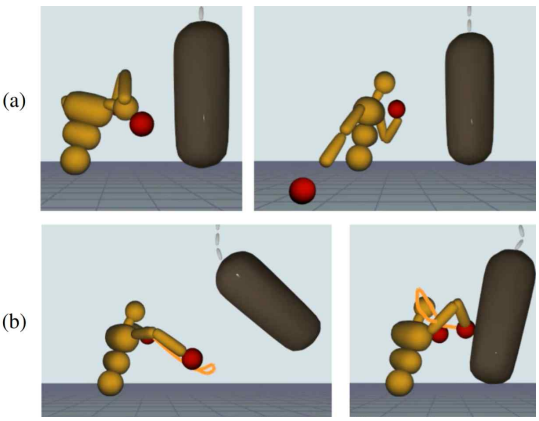



Fig. 12. Failure case of our work.

물리 엔진의 정확도와 보상 조건 설정의 정밀도를 높일 필요가 있었다.

게임 분야에서 캐릭터 애니메이션과 물리 사뮬레이션을 함께 사용하는 경우는 Ragdoll 형태나 환경과 상호작용 정도의 제한된 용도로 사용되고 있다. 하지만 강화학습의 연속 행동 공간(continuous action space)을 다루는 기법을 통해서 수동적인 반응뿐 아니라 능동적인 행동도 처리할 수 있는 가능성이 있다고 생각한다.

향후 연구에서는 A2C와 다른 강화학습 알고리즘을 적용해 비교해보고 인공신경망을 경량화해 실시간 캐릭터 애니메이션에 적용할 계획이다. 또한 물리 엔진과 캐릭터 애니메이션에서 강화학습을 접목시킬 수 있는 기법을 연구할 계획이다.

## REFERENCE


[1] J.A. Boyan and A.W. Moore, "Generalization in Reinforcement Learning: Safely Approximating the Value Function," *Advances in Neural Information Processing Systems*, pp. 369-376, 1995.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, et. al, "Human-level Control through Deep Reinforcement Learning," *Nature*, Vol. 518, No. 7540, pp. 529-533, 2015.

[3] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. van den Driessche, et. al., "Mastering the Game of Go with Deep Neural Networks and Tree Search," *Nature*, Vol. 529, No. 7587, pp. 484-489, 2016.

[4] X.B. Peng, G. Berseth, and M. van de Panne, "Terrainadaptive Locomotion Skills Using Deep Reinforcement Learning" *ACM Transactions on Graphics*, Vol. 35, No. 4, pp. 81:1-81:12, 2016.

[5] S. Hong, "Physics-based Character Motion Research," *Media Storytelling*, Vol. 1, pp. 66-79, 2014.

[6] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust Region Policy Optimization," *Proceedings of the 32nd International Conference on Machine Learning*, pp. 1889-1897, 2015.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *Computing Research Repository*, arXiv:1707.06347, 2017.

[8] K. Yin, K. Loken, and M. Van de Panne, "Simbicon: Simple Biped Locomotion Control," *ACM Transactions on Graphics*, Vol. 26, No. 3, pp. 1-10, 2007.

[9] T. Kwon and J. Hodgins, "Control Systems for Human Running Using an Inverted Pendulum Model and a Reference Motion Capture Sequence," *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 129-138, 2010.

[10] M. De Lasa, I. Mordatch, and A. Hertzmann, "Featurebased Locomotion Controllers," *ACM Transactions on Graphics*, Vol. 29, No. 4, pp.131:1-131:10, 2010.

[11] S. Ha and C.K. Liu, "Iterative Training of Dynamic Skills Inspired by Human Coaching Techniques," *ACM Transactions on Graphics*, Vol. 34, No. 1, pp.1:1-1:11, 2014.

[12] Y. Lee, S. Kim, and J. Lee, "Data-driven Biped

[13] V.B. Zordan and J.K. Hodgins, "Motion Capturedriven Simulations that Hit and React," *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 89-96, 2002.

[14] J. Hwang, I. Suh, and T. Kwon, "Editing and Synthesizing Two-character Motions Using a Coupled Inverted Pendulum Model," *Computer Graphics Forum*, Vol. 33, No. 7, pp. 21-30, 2014.

[15] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, 1998.

[16] Y.S. Sung and K.E. Cho, "Q-learning Using Influence Map," *Journal of Korea Multimedia*

*Society*, Vol. 9, No, 5, pp. 649-657, 2006.

[17] H. van Hasselt and M.A. Wiering, "Reinforcement Learning in Continuous Action Spaces," *Proceeding of 2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pp. 272-279, 2007.

[18] J.A. Bagnell and J. Schneider, "Covariant Policy Search," *Proceeding of International Joint Conference on Aritifical Intelligence*, pp. 1019-1024, 2003.

[19] H.J. Kappen, "Path Integrals and Symmetry Breaking for Optimal Control Theory," *Journal of Statistical Mechanics: Theory and Experiment*, Vol. 2005, No. 11, p. P11011, 2005.

[20] J. Peters, K. Mülling, and Y. Altun, "Relative Entropy Policy Search," *Proceeding of Association for the Advancement of Artificial Intelligence Atlanta*, pp. 1607-1612, 2010.

[21] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," *Proceeding of International Conference on Machine Learning*, pp. 1928-1937, 2016.

[22] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic Policy Gradient Algorithms," *Proceedings of the 31st International Conference on Machine Learning*, pp. 387-395, 2014.

[23] 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," *Computing Research Repository*, arXiv:1509.02971, 2015.

[24] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, et. al., "Emergence of Locomotion Behaviours in Rich Environments," *Computing Research Repository*, arXiv: 1707.02286, 2017.

[25] J. Tan, Y. Gu, C. K. Liu, and G. Turk, "Learning Bicycle Stunts," *ACM Transactions on Graphic*, Vol. 33, No. 4, pp. 50:1-50:12, 2014.

[26] Y. Wang, E. Li, F. Wang, and B. Xu, "A Virtual Character Learns to Defend Himself in Sword Fighting Based on Qnetwork," *Procceding of 2016 IEEE 28th International Conference on Tools with Artificial Intelligence*, pp. 291-298, 2016.

[27] L. Liu and J. Hodgins, "Learning to Schedule Control Fragments for Physics-based Characters Using Deep q-Learning," *ACM Transactions on Graphic*, Vol. 36, No. 3, pp. 29: 1-29:14, 2017.

[28] J. Won, J. Park, K. Kim, and J. Lee, "How to Train Your Dragon: Example-guided Control of Flapping Flight," *ACM Transactions on Graphic*, Vol. 36, No. 4, pp. 198:1-198:13, 2017.

[29] X.B. Peng, G. Berseth, K. Yin, and M. van de Panne, "Deeploco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning," *ACM Transactions on Graphics*, Vol. 36, No. 4, pp. 41:1-41:13, 2017.

[30] H.G. Armstrong, "Anthropometry and Mass Distribution for Human Analogues," *Military Male Aviators*, Vol. 1, 1988.

[31] M. Cheraghi, H.A. Alinejad, A.R. Arshi, and E. Shirzad, "Kinematics of Straight Right Punch in Boxing," *Annals of Applied Sport Science*, Vol. 2, No. 2, pp. 39-50, 2014.

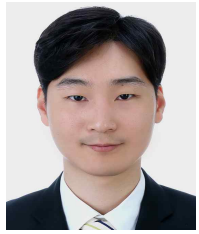
[32] S.J. Kim and B.H. Woo, "Biomechanical Analysis of Left and Right Hook Type on Trunk Motion in Boxing," *The Korea Journal of Sports Science*, Vol. 22, No. 6, pp. 1519-1529, 2013.

박 현 준

2011년~현재 동국대학교 컴퓨터공학전공 학사
관심분야 : 컴퓨터 그래픽스, 강화학습, 캐릭터 애니메이션, 증강현실

최 위 동

2018년 동국대학교 컴퓨터공학전공 석사
관심분야 : 컴퓨터 그래픽스, 인공지능, 캐릭터 애니메이션, 물리 시뮬레이션



장 승 호

2015년 동국대학교 컴퓨터공학전공 학사
2017년 동국대학교 컴퓨터공학전공 석사
2017년~현재 동국대학교 컴퓨터공학전공 박사과정
관심분야 : 컴퓨터 그래픽스, 인공지능, 3차원 모델링, 3D 프린터

홍 정 모

2000년 학국과학기술원 기계공학전공 학사
2002년 한국과학기술원 기계공학전공 석사
2005년 고려대학교 전산학전공 박사
2005년~2007년 Stanford University(USA) 연구원
2008년~현재 동국대학교 컴퓨터공학과 부교수
관심분야 : 컴퓨터 그래픽스, 컴퓨터 애니메이션, 강화학습