한국컴퓨터그래픽스학회
Korea Computer Graphics Society Vol. 30, No. 4, P. 1~10

강화학습과 Motion VAE 를 이용한 자동 장애물 충돌 회피 시스템 구현

사정 O 구태홍 1 권태수*
한양대학교 컴퓨터소프트웨어학과
wangshengyufei@naver.com, gestoru@gmail.com,taesoobear@gmail.com

An Auto Obstacle Collision Avoidance System using Reinforcement Learning and Motion VAE

Zheng Si O Taehong Gu 1 Taesoo Kwon*
Department of Computer Science, Hanyang University, South Korea

요 약

컴퓨터 애니메이션 및 로보틱스 분야에서 장애물을 회피하면서 목적지에 도착하는 것은 어려운 과제이다. 특히 이동 경로의 계획과 적절한 동작의 계획을 동시에 수행하는 것은 기존에 많이 다루어지지 않은 연구분야이다. 최근 연구자들은 데이터 기반 생성 모델인 VAE(Variational Auto-Encoder)를 활용하여 캐릭터 모션을 생성하는 문제를 활발하게 연구해왔다. 본 연구에서는 VAE 생성 모델을 동작 공간으로 확장하고, 잠재 공간에서 강화학습을 적용한 MVAE 모델을 활용한다[1]. 이 접근 방식을 통해 학습된 정책을 사용하면 에이전트는 고정된 장애물과 움직이는 장애물을 모두 피하면서, 자연스러운 움직임으로 목적지에 도달할 수 있다. 캐릭터는 무작위 방향으로 이동하는 장애물을 견고하게 회피할 수 있었고, 기존 연구보다 성능이 뛰어나고 학습 시간도 단축됨을 실험적으로 보였다.

Abstract

In the fields of computer animation and robotics, reaching a destination while avoiding obstacles has always been a difficult task. Moreover, generating appropriate motions while planning a route is even more challenging. Recently, academic circles are actively conducting research to generate character motions by modifying and utilizing VAE (Variational Auto-Encoder), a data-based generation model. Based on this, in this study, the latent space of the MVAE model is learned using a reinforcement learning method[1]. With the policy learned in this way, the character can arrive its destination while avoiding both static and dynamic obstacles with natural motions. The character can easily avoid obstacles moving in random directions, and it is experimentally shown that the performance is improved, and the learning time is greatly reduced compared to existing approach.

키워드: Motion VAE, 강화학습, 장애물 회피, 캐릭터 제어, 캐릭터 애니메이션
Keywords: Motion VAE, reinforcement learning, obstacle avoidance, character control, character animation

1. 서론

강화 학습은 에이전트가 환경과 반복적으로 상호작용하는 시행 착오 경험을 통해 누적 보상을 최대화 하는 정책을 학습하는 기법이다. 강화학습은 우수한 제어 정책을 얻기 위해 많은 시행착오 샘플이 필요하다는 단점에도 불구하고, 다양한 종류의 태스크와 환경에 대한 캐릭터 제어 정책을 환경에 대한 정교한 정보 없이도 얻을 수 있다는 장점 때문에 많이 사용되고 있다.

최근 몇 년간 심층 강화 학습 기반 접근 방식은 주로 물리적으로 시뮬레이션되는 캐릭터를 제어하는 문제에서 좋은 결과를 보여왔다. 2020 년에 나온 Ling 등의 논문은 강화학습을 MotionVAE(이하 MVAE)라고 명명된 데이터 기반의 생성 모델에도 효과적으로 적용시킬 수 있다는 것을 보였다[1]. 이 연구는 미리 학습된 MVAE 모델이 주어지면,

*corresponding author: Taesoo Kwon/Hanyang University(taesoobear@gmail.com)

Received : 2024.04.08./ Review completed : 1st 2024.05.08. / Accepted : 2024.06.17.
DOI : 10.15701/kcgs.2024.30.4.1
ISSN : 1975-7883(Print)/2383-529X(Online)

MVAE 모델의 인코더는 폐기하고, 잠재 공간 디코더만 가지고 강화 학습 과정을 진행한다. 즉, 강화학습 정책의 action space 는 MVAE 모델의 잠재 공간이고 디코더는 강화학습 정책의 action 출력을 받아서 대응되는 다음 프레임의 포즈를 생성한다. 즉, 예제 동작의 표현 공간에서 정책의 탐색이 이루어지기 때문에 자연스러운 동작의 생성이 가능하다.

본 연구의 목표는 캐릭터가 장애물이 많고 폐쇄된 환경에서도 환경을 스스로 감지하여 목적지에 도착할 수 있도록 강화학습 정책을 학습하는 것이다. 이는 기존 MVAE 논문에서 다루지 않은 태스크이다. 기존에 보행동작 및 군중 동작 생성에 강화학습을 활용한 기존 연구들이 있었고[2, 3], 본 연구도 이들과 유사한 보상 함수를 사용하지만, 이들과 달리 제안된 방법은 MVAE 잠재 공간에서 학습을 수행한다는 차이점이 있다. 본 논문의 컨트리뷰션은 다음과 같이 요약 가능하다.

- 기존 MVAE 논문에서 다루지 않았던 복잡한 환경에서 MVAE 모델의 생성 능력을 실험적으로 검증하였다.
- 고정된 장애물 환경과 이동하는 장애물 환경에서 별도의 학습이 필요한 DeepLoco[2] 논문과 달리, 우리의 방법은 고정된 장애물 환경에서 한 번만 학습하고, 학습된 정책은 추가 학습 없이 무작위로 움직이는 장애물이 있는 환경에 적용될 수 있다.

2. 관련 연구

2.1 장애물 회피

장애물 자동 회피에 대한 연구는 컴퓨터 그래픽스 및 로보틱스 분야에서 많이 진행되어 왔다. 최근 저출산 및 노령화 문제 등으로 인해 지능형 로봇에 대한 수요가 급증하고 있다. 이에따라 학계와 산업계에서는 로봇 움직임 플랜닝 시스템과 장애물 회피 시스템을 연구하였다. 1990 년에 V. Lumelsky 연구팀은 Bug algorithms 이라 불리는 장애물 회피 문제에 대한 기본적인 경로 탐색 기법을 발표하였다[4, 5]. 이 방법은 로봇이 시작점에서 목표를 향해 직선으로 이동 중 장애물과 만나면 먼저 장애물의 가장자리를 따라 회피 후 다시 직선 경로를 찾아 이동을 재개한다. O. Khatib 연구팀은 합력을 계산하여 장애물을 회피하는 Potential Field 방법을 발표하였다[6]. Borenstein 연구팀은 Vector Field Histogram 방법을 발표하였다[7]. 로봇 주변의 환경의 정보를 수집하여 히스토그램을 구축하고 장애물의 밀도를 판단하여 이동 방향을 결정하는 방법이다.

이와 같은 전통적인 규칙기반의 방법 외에도, 최근에는 인공신경망을 사용하여 장애물 회피 및 경로 계획 문제를 해결하는 연구가 활발하다. 특히 대표적인 방법은 강화학습으로 마르코프 결정 과정(MDP)을 해결하는 알고리즘이고, 연속적으로 결정을 내리는 문제에 유용하다. 최적의 정책을 계산하여 그 문제에서 정의된 리턴값을 최대화 시킨다. 강화 학습은 학습된 정책이 현재 상태 또는 몇 단계 앞선 시간 단계에 대해 일관되게 최적의 액션을 출력할 수 있기 때문에 장애물 회피 및 경로 계획 문제에 특히 적합한 문제 해결 방법이다. Volodymyr Mnih 와 David Silver 등 연구자들이 몇몇의 도전적인 문제에서 심층 강화학습의 유용함을 증명하였다[8, 9]. 이재동 등은 강화학습과 속도 기반 모델을 결합하여 crowd simulation 에서 성공적인 결과를 보였다[3]. Xuebin Peng 등은 강화학습과 물리 기반 보행 프레임워크와 결합하여 장애물을 피하면서 목적지에 도착할 수 있게 해 주는 정책을 학습하였다[2].

2.2 강화학습

시뮬레이션 되는 캐릭터들을 제어하는 컨트롤러들을 발전시키기 위해 사용되는 많은 종류의 최적화 기술들이 강화 학습을 기초로 활용하고 있다. 가치 반복방법들(value iteration methods)은 모션 클립들을 적절한 순서로 배열하기 위한 운동학(kinematics)기반 제어기들을 발전시키기 위해 사용되어 왔다 [10, 11]. 또한 동역학(dynamics)을 사용한 제어기 연구들에서도 이와 비슷한 접근 방법이 연구되었다 [12, 13]. 이러한 강화 학습에 최근 심층 신경망이 적용되며 더 다양한 도전적인 과제들을 수행할 수 있게 되었다 [14, 15]. 최근에는 생체 역학 모델을 활용하여 자연스러운 움직임을 생성하는 연구도 진행되었다 [16].

3. 시스템 개요

Figure 1 은 우리가 개발한 프레임워크의 전체 구조이다.각 시간 단계에서 캐릭터는 환경과 상호 작용하고 자신의 행동과 환경 상태에 따라 보상을 받는다. 그림 왼쪽의 정책(Policy)는 강화학습을 사용하여 학습된다. 캐릭터 모션의 필수 특징을 인코딩하는 잠재 벡터는 각 시간 단계마다 계산되어 디코더로 전달된다. 그림 가운데의 디코더는 잠재 벡터 z 와 지금 타임스텝의 캐릭터의 포즈를 입력 받아 다음 타임스텝의 캐릭터 포즈를 출력한다. 4 절, 5 절에서는 MVAE 와 강화학습 정책에 대해 자세히 설명하겠다.

4. Motion VAE

딥러닝 기술이 점점 발전됨에 따라 학계에서 VAE(Variational Autoencoder)와 GAN(Generative Adversarial Network) 등 다양한 생성 모델에 대해서 활발하게 연구해 왔다. 이 논문에서 사용한 생성 모델은 Motion VAE 이다[1]. 이 모델은 컴퓨터 애니메이션 분야에서 모션 캡쳐 데이터를 처리 및 학습하기 위해서 2020 년에 발표된 모델이다. 본 연구는 Motion VAE 모델을 low-level 동작 생성기로 사용한다.

Figure 2 은 VAE 에 대한 설명이다. 이 모델은 학습할 때 인풋을 받고 인코더로 인풋에 지닌 정보를 압축하여 잠재 공간에서 평균값과 분산을 만들고 다시 디코더로 잠재 공간에서 평균값과 분산을 가지고 샘플링한 값에 대해서 압축을 풀어서 아웃풋을 출력한다. 이 때 Loss 는 인풋과 아웃풋의 차이를 최소화하고 인코더가 출력하는 평균값과 분산을 정규분포에 가까워지도록 KL divergence 를 최대화하는 것이다. 인풋과 아웃풋의 차이가 작을수록 KL divergence 가 크고, 인풋 데이터셋에 대한 네트워크의 피팅이 잘 된다고 할 수 있다. 생성 단계에서는 앞의 인코더 부분은 폐기하고 잠재 공간과 디코더만 남긴다. 잠재 공간의 임의의 벡터(잠재벡터)를 입력하면 디코더는 그에 해당되는 자세를 출력한다. 이처럼 유한의 데이터셋을 미리 학습해 놓으면, 연속된 공간에서 원래 데이터셋에 없는 새로운 자세도 생성할 수 있는 것이 VAE 생성 모델의 특징이다.

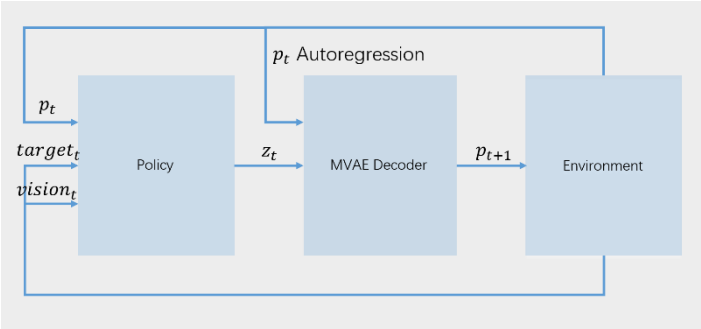


Figure 1: Using DRL, we can learn a policy that computes the latent variable z at every step to guide the character. In each time step, the environment uses p_t generated by the decoder to integrate the root positions and computes relevant task information $target_t$ and $vision_t$. The policy computes the action from p_t, $target_t$ and $vision_t$ which is fed as z_t back into the decoder.

Figure 3 은 Motion VAE(MVAE) 의 전체적인 구조로, 일반적인 VAE 와 유사하지만 세부 사항에서 몇 가지 차이점이 있다. 먼저 MVAE 의 인코더의 경우 지난 프레임의 포즈 데이터와 이번 프레임의 포즈 데이터를 인풋으로 받고 인코딩한다. 즉 각 프레임마다 포즈의 전이 정보를 인코딩하는 것이다. 또 다른 차이점은, 일반적인 VAE 의 디코더는 잠재 공간만 디코딩하는 반면에 MVAE 의 디코더는 지난 프레임에 예측한 포즈도 같이 이용한다는 점이다. 즉, 기존 포즈도 잠재 벡터와 같이 디코더에 입력으로 들어가서 예측 작업에 활용된다는 점에서 MVAE 는 auto-regressive conditional variational autoencoder 라고도 불린다.

MVAE 는 모션 캡쳐 데이터를 이용하여 감독 학습 방법으로 학습한다. MVAE 는 다음 프레임에 나올 포즈들의 분포를 모델링 한다. 디코더 입력으로 들어간 지난 프레임의 포즈가 똑같아도 샘플링된 잠재 벡터가 다르면 예측 결과가 달라진다. 본 논문에서는 모션 캡쳐 데이터를 활용하여 미리 학습된 MVAE 네트워크를 디코더로 사용한다. 즉, 본 연구는 MVAE 논문에서 사용한 캐릭터 모델과 MVAE 디코더 네트워크를 그대로 사용하여 연구를 진행하였다. 장애물과의 충돌을 회피할 수 있는 캐릭터 자동 회피 시스템을 구현하기 위하여 본 연구는 강화학습 방법을 사용한다. MVAE 디코더를 시뮬레이터로 활용하여 시행착오 데이터를 취득하고, 이를 통해 태스크 목적에 맞는 적절한 잠재 벡터를 출력할 수 있는 정책을 학습한다.

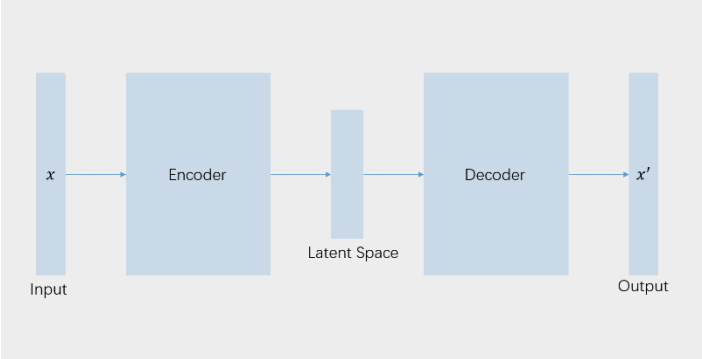


Figure 2: The basic scheme of a variational autoencoder. The model receives x as input. The encoder compresses it into the latent space. The decoder receives as input the information sampled from the latent space and produces x' as similar as possible to x.

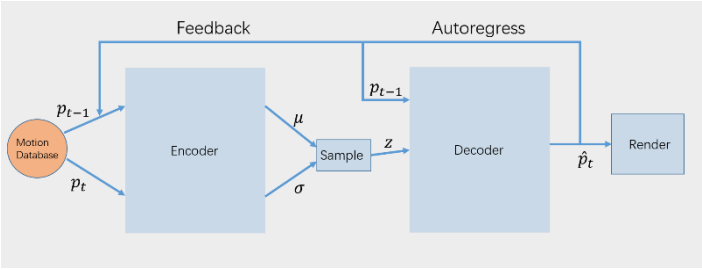


Figure 3: The conditional VAE has two parts. The encoder takes past (p_{t-1}) and current (p_t) pose as input and outputs both μ and σ, which is then used to sample a latent variable z. The decoder uses p_{t-1} and z to reconstruct p_t.

5. 방법 및 구현

캐릭터를 많은 고정된 장애물이 있고 폐쇄된 다양한 환경에 투입하여 정책을 학습한다(Figure 4). 학습시 목적지는 고정되어 있고, 캐릭터의 초기 위치는 랜덤하게 설정된다. 학습이 완료되고 학습된 정책을 사용할 때는, 학습 시와 달리 캐릭터의 초기 위치를 목적지와 멀리 떨어진 위치에 고정 시키고, 캐릭터의 진행 방향을 랜덤으로 설정한다. 이렇게 학습된 정책은 장애물이 움직이는 환경에서도 그대로 사용할 수 있다.

5.1 캐릭터 모델

본 논문에서는 66 자유도의 캐릭터를 사용한다. 포즈 데이터를 표현할 때 먼저 루트 관절의 위치를 바닥에 투영시켜 x, y 값만 얻는다. 또한 루트 관절의 방향을 바닥에 투영시켜 캐릭터가 바라보고 있는 방향(케릭터 로컬 좌표계의 x 축)과 글로벌 x 축의 각도를 계산해서 페이싱(facing) 방향 a 를 얻는다. 이 x, y, a 값을 이용해서 캐릭터의 선속도와 각속도($\dot{r}^x, \dot{r}^y, \dot{r}^a \in R$)를 계산할 수 있다. 나머지 관절의 위치($j^p \in R^3$)와, 속도($j^v \in R^3$)는 모두 루트 공간을 기준으로 표현한다. 관절의 회전값($j^r \in R^6$)은 그들의 forward vector 와 upward vector 로 6 차원으로 표현하고 루트 좌표계에 상대적으로 표현한다. 본 논문에서는 포즈 p 를 다음과 같이 정의한다.

$$p = (\dot{r}^x, \dot{r}^y, \dot{r}^a, j^p, j^v, j^r).$$

이 포즈 데이터는 강화학습을 진행할 때 observation space 의 일부로 사용된다.

5.2 시뮬레이션 환경

5.2.1 정지된 장애물 환경

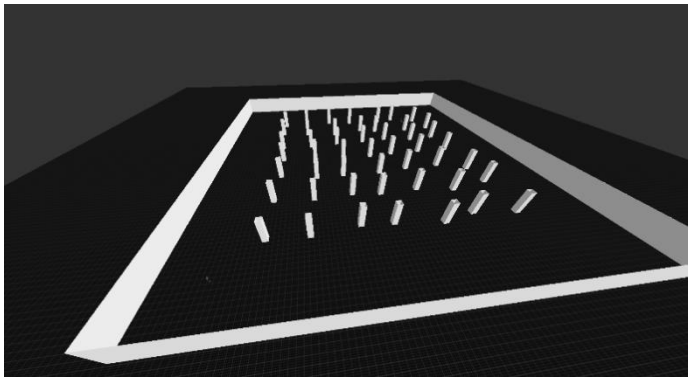


Figure 4: A closed environment with static obstacles

Figure 4 와 같이 본 논문에서는 길이 300 미터, 너비 300 미터인 폐쇄된 환경 내부에서 캐릭터 시뮬레이션을 진행한다. 이 가운데 27~33 미터의 랜덤 간격을 두는 장애물 49 개를 배치하였다. 모든 장애물의 크기는 동일하게 변의 길이를 4 미터로 설정한다. 이 예제에서 케릭터의 출발 위치는 그림의 오른쪽 위에 있다. 목적지는 왼쪽 아래에 위치하고 Figure 4 에서 파란색으로 표시하였다.

5.2.2 움직이는 장애물 환경

정지된 장애물과 똑같은 초기 위치를 가지고 있고, 시뮬레이션을 시작할 때 랜덤한 시작 방향과 0~0.15$m/timestep$의 랜덤한 속도로 시작한다.

5.3 제어 정책의 학습

5.3.1 강화학습

MVAE 의 잠재 공간을 동작 공간으로 사용하여 강화학습을 진행할 수 있다(Figure 1). 강화학습 진행 시 매 타임스텝 t 마다 에이전트가 액션 a_t 하나를 선택하여 환경 상태 s_t와 상호작용 한다. s_t에서 액션 a_t를 실행한 후, 환경 상태가 s_{t+1} 되고 에이전트가 환경에서 보상 $r_t = r(s_t, a_t)$ 을 피드백으로 받는다. 강화학습 진행 과정에서 에이전트는 신경망 정책 $\pi_\theta(a \vee s)$으로 동작 a_t를 계산한다. 신경망 정책 $\pi_\theta(a \vee \cdot)$ 는 지금 정책에서 동작 a 의 확률 밀도이다. 강화학습의 목적은 다음 공식을 최대화시킬 수 있는 네트워크 파라미터를 찾는 것이다:

$$J_{RL}(\theta) = E\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$$

여기서 $\gamma \epsilon [0,1)$ discount factor 로 보상의 합이 수렴할 수 있도록 설정한다. 본 논문에서는 이 최적화 문제를 푸는 데 proximal policy optimization(PPO) 알고리즘을 사용한다. PPO 알고리즘의 구현은 공개된 버전[17]을 사용한다.

5.3.2 제어기 네트워크

제어 정책은 히든 레이어 2 개가 있는 신경망이다. 히든 레이어에 256 개의 히든 유닛이 있고 활성화함수는 ReLU 를 사용한다.이 연구에서 다음 공식에 의해서 시간에 따라 감소되는 학습율을 사용한다:

$$max(1{,}3 * 0.99^{iteration}) \times 10^{-5}$$

지수 형태로 학습율을 일정량까지 감소시키는 것은 정책 네트워크의 학습의 안정성에 큰 도움이 된다. 학습 데이터를 수집하기 위해서는 100 개의 쓰레드로 병렬 시뮬레이션을 실행한다. 매 시뮬레이션의 최대 타임 스텝은 1200 으로 세팅한다. 정책 네트워크와 벨류 네트워크는 1000 샘플의 미니 배치로 업데이트한다. AMD R9 7950X 와 Nvidia GeFore GTX 1080 에서 학습 진행 시, 정책 네트워크가 완전히 수렴할 때까지 걸린 시간은 총 72 시간이다.

5.3.3 제어 정책의 상태 공간

우리의 정책 네트워크는 다음과 같은 상태 공간을 사용한다

$$s_t = posedata(condition)_t \cdot vision_t \cdot target_t$$

$posedata(condition)_t$ 은 앞서 정의한 267 차원의 캐릭터의 pose t 를 의미한다. 이 값은 실시간으로 계산되며 MVAE 네트워크의 출력에서 전달받을 수 있다. $vision_t$는 Figure 5 와 같이 캐릭터의 간단한 시각 시스템을 의미한다. 이 시각 시스템은 60 차원이며 180 도의 부채꼴 범위에서 레이를 60 방향을 발사한다. 매 레이는 최대 50 미터까지 갈

수 있으며 장애물이나 벽과 부딪치게 되면 더 이상 갈 수 없고 그 자리에서 멈추게 되며 장애물과의 거리를 반환한다. 레이가 장애물과 부딪치지 않으면 50 을 반환한다.캐릭터의 에이전트는 이 간단한 시각 시스템을 통해서 전방 및 좌우 50 미터 이내의 상황을 파악한다. $target_t$ 는 목적지의 위치를 나타내는 2 차원 벡터이다. 캐릭터의 로컬 좌표계에서 표현된다. '·'는 텐서의 연결을 의미한다.

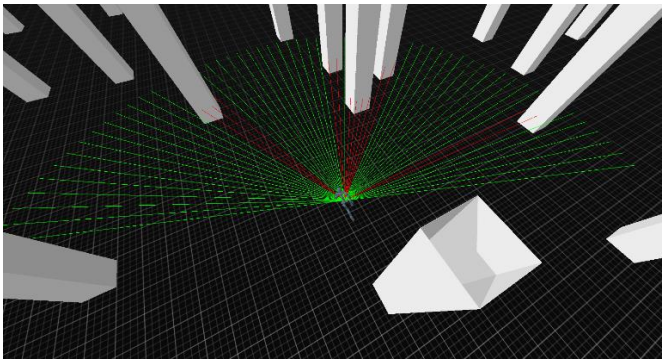
우리의 상태 공간의 총 차원 수는 329 이다. 이 상태 공간에 있는 모든 값이 매 타임 스텝 계산되며 정책 네트워크의 인풋으로 사용된다.

5.3.4 보상 함수

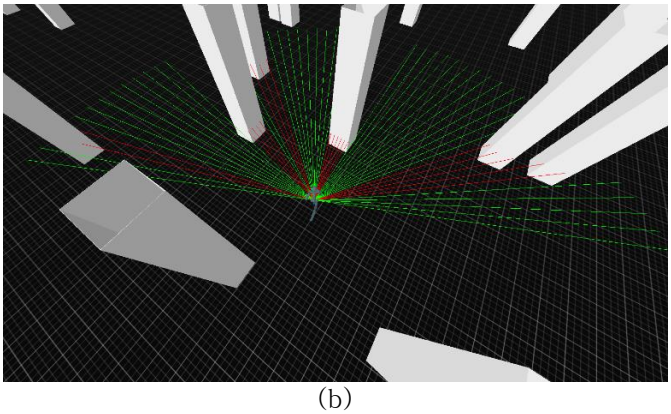
학습 시 보상 함수를 다음과 같이 정의한다:

$$r = (\sim cto) \times r_t + cto \times r_o + r_e + r_{tic}$$

cto는 장애물과의 거리 척도를 바이너리 값으로 나타낸다. 본 연구에서 이 한계치를 15 미터로 지정한다. 즉 캐릭터가 임의의 장애물과 15 이하의 거리를 유지하게 되면 이 c 값을 1 로 세팅하고 아니면 0 으로 지정한다.



(a)



(b)

Figure 5: Vision system

$$r_t = \left(\sqrt{(p_{root} - p_{target})^2}\right)_{t-1} - \left(\sqrt{(p_{root} - p_{target})^2}\right)_t$$

r_t는 루트의 위치가 목적지와의 거리의 변화를 의미한다. 지난 타임 스텝보다 목적지와의 거리가 가까워지면 양수, 멀어지면 음수가 된다.

r_o는 임의의 장애물과의 거리가 15 미터 이하가 되면 작동하기 시작한다. 이 상태에서 그 장애물과 가까워지면 음수, 멀어지면 양수가 된다. 학습 시 장애물들이 모두 정지되어 있고, 장애물과 장애물 간의 간격은 최소 30 미터이기 때문에 동일 시간에 캐릭터와의 거리가 15 미터 이하인 장애물은 한개밖에 없다.

r_e는 effort penalty 이다. 이 값은 다음과 같이 계산된다.

$$E = (\dot{r}^x)^2 + (\dot{r}^y)^2 + (\dot{r}^a)^2 + \frac{1}{J}\sum_j^J \|j^v\|^2 .$$

기존 MVAE 논문에서는 캐릭터의 모션이 더 자연스럽고 부드럽게 나오게 하기 위해서 보상 함수에 이 effort penalty 항목을 추가해야 한다고 언급했다. RL 문제에서 에너지 페널티가 보상 함수에 들어가면 최적화 과정에서 낮은 에너지의 솔루션을 찾을 수 있다는 것이다. 에너지가 낮으면 그만큼 모션이 시각적인 측면에서 사람의 움직임에 더 가까워지고 자연스러울 것이다. 하지만 본 연구에서는 장애물을 피하는 목적을 달성하기 위해서 학습 시 이 항목을 제거하였다. 자세한 설명은 6.3 절에 기술한다.

r_{tic}는 캐릭터가 목적지의 주변에 도착하게 되면 이 보상을 획득하게 된다. 본 연구에서 주변의 판정을 8 미터로 세팅하며 보상 값을 20 으로 고정 시킨다.

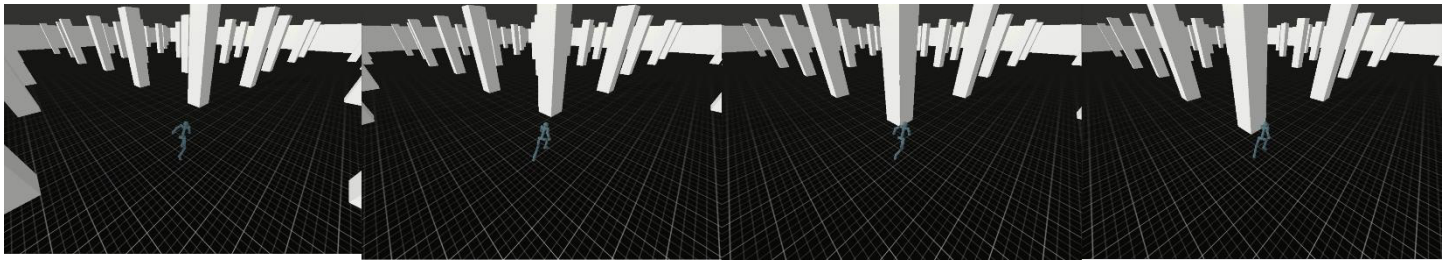
6. 실험 및 평가

본 논문에서는 먼저 정지된 장애물 환경에서 학습을 진행한다. 보상의 합이 수렴하고 학습이 충분히 된 다음에 학습된 정책의 실제 성능을 그 환경에서 테스트한다. 정지된 장애물 환경에서의 실험이 끝나고 움직이는 장애물 환경에서 학습된 정책을 그대로 사용하여 테스트를 진행한다. 구현 및 테스트를 진행할 때 pybullet 을 사용하였고 정책의 학습을 진행할 때 pytorch 를 사용하였다.

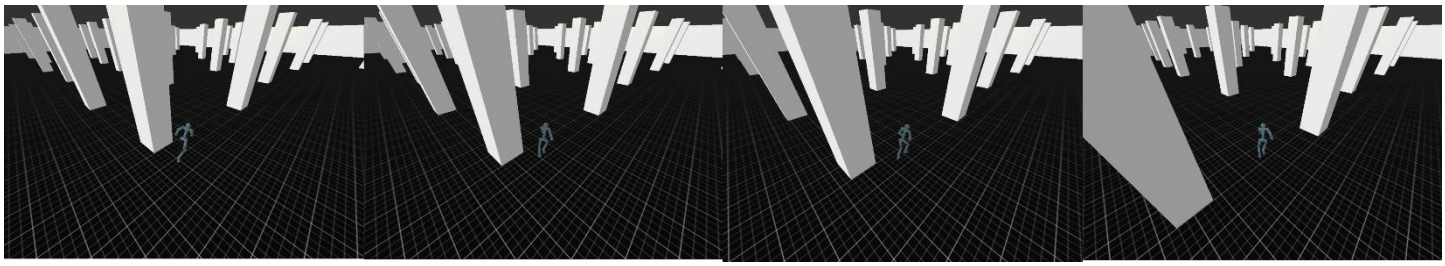
6.1 정지된 장애물 환경

본 논문에서는 정지된 장애물 환경에서 학습을 진행할 때 deepmimic[18] 논문에서 소개한 ET (Early Termination) 방법과 ISD (Initial State Distribution) 방법을 사용하였다.

ET 방법은 캐릭터가 장애물과 부딪치게 되면 바로 그



(a) The character is approaching an obstacle and trying to avoid it



(b) The character successfully avoids the obstacle and continues towards the destination

Figure 6: Character simulation snapshots in static obstacles environment. The policy we trained allows the character to find a path from start position to destination position without colliding with any obstacle.

에피소드를 종료 시키는 것이다. 강화학습의 학습 효율은 학습 데이터의 퀄리티에 달려 있다. 캐릭터가 장애물과 부딪치게 되면 캐릭터의 팔이나 다리가 장애물에 파고 들면서 불필요한 데이터가 생기기 때문에 이 때 에피소드를 종료 시키는 것이 학습 효율에 큰 도움이 된다. 본 연구 DeepLoco [2]와 달리 운동학 기반 모션 생성이기 때문에 충돌이 일어나게 되면 즉시, 에피소드를 종료 시켜야 한다. 반면 DeepLoco 는 동역학에 기반하기 때문에 충돌이 일어나도 캐릭터가 넘어지기만 않으면 그 에피소드를 종료시킬 필요가 없다.

ISD 방법은 캐릭터의 초기 위치 분포를 의미한다. 정책을 학습할 때 캐릭터의 초기 위치를 임의의 장애물 주변에서 샘플링하며, 캐릭터의 초기 진행 방향도 0 과 π 사이의 랜덤 값으로 초기화 시킨다. 학습된 정책을 테스트할 때는 캐릭터의 초기 위치를 고정 위치로 설정한다. 하지만 학습 과정에서 캐릭터의 초기 위치를 고정 위치로 설정하면, 에이전트는 고정 위치 주변에서만 경로를 찾을 수 있고, 시작 위치부터 목적지까지의 전체 경로를 찾아낼 수 없을 것이다. 강화학습에서 전체 공간에 대한 탐색이 잘 이루어지지 않으며 학습이 잘 진행되지 않거나 비효율적으로 진행된다. 학습 시 캐릭터의 초기 위치를 전체 공간에 고르게 배치하면 캐릭터의 출발위치에 관계 없이, 임의의 시작 위치부터 목적지까지의 경로를 쉽게 찾아낸다.

정지된 장애물 환경에서 학습할 때 보상의 합이 수렴할 때까지 총 72 시간의 학습 시간이 걸렸다. 보상의 합의 중간값과 평균값의 곡선이 Figure 8 에서 확인할 수 있다.

이렇게 학습함으로써 캐릭터가 정지된 장애물 환경에서 시작 위치부터 장애물을 피하면서 목적지에 도착할 수 있다. Figure 6 에서 결과를 확인할 수 있다.

6.2 움직이는 장애물 환경

정지된 장애물 환경에서 학습한 정책을 그대로 사용하여 움직이는 장애물 환경에 적용시킬 수 있다. Figure 7 에서 결과를 확인할 수 있다. 캐릭터가 고정 시작 위치에서 출발하면서 장애물도 동시에 랜덤 방향으로 움직이기 시작한다. 도중에 장애물이 캐릭터 앞에 들어오게 되면 캐릭터가 장애물들 사이에서 경로를 찾아서 유연하게 통과할 수 있다.

6.3 정책의 성능

본 연구 진행 시 보상 함수에 effort penalty 항목은 추가하지 않는 것이 더 좋은 장애물 회피 성능을 얻을 수 있다는 것을 발견하였다. 일반적으로 사람이나 동물은 걷다가 앞에 장애물이 나타나면 적절한 방향으로 가속해서 장애물을 회피한다. 에이전트도 사람과 유사하게 가속을 하는 액션을 선택하는 경향이 있었고, effort penalty 항목은 이를 방해하는 방향으로 동작하여, 장애물을 피해야 할 때 느리게 반응하여 잘 못 피하는 상황을 만들었다. 장애물이 나타났을때 캐릭터가 이를 피하려고 가속 하면 에너지 페널티를 받을 것이고, 페널티를 받지 않으려고 가속하지 않고 그대로 가면 장애물과 충돌이 일어나서 해당 에피소드가 끝나고 더 이상 보상을 받지 못하는 문제였다. 결과적으로 캐릭터 모션의 품질을 조금 희생하더라도 장애물 회피 성능을 확보하기 위하여 effort penalty 항목은 사용되지 않도록 설정하였다.

물론 이 항목을 사용하지 않으면 우리는 원하는 동작보다

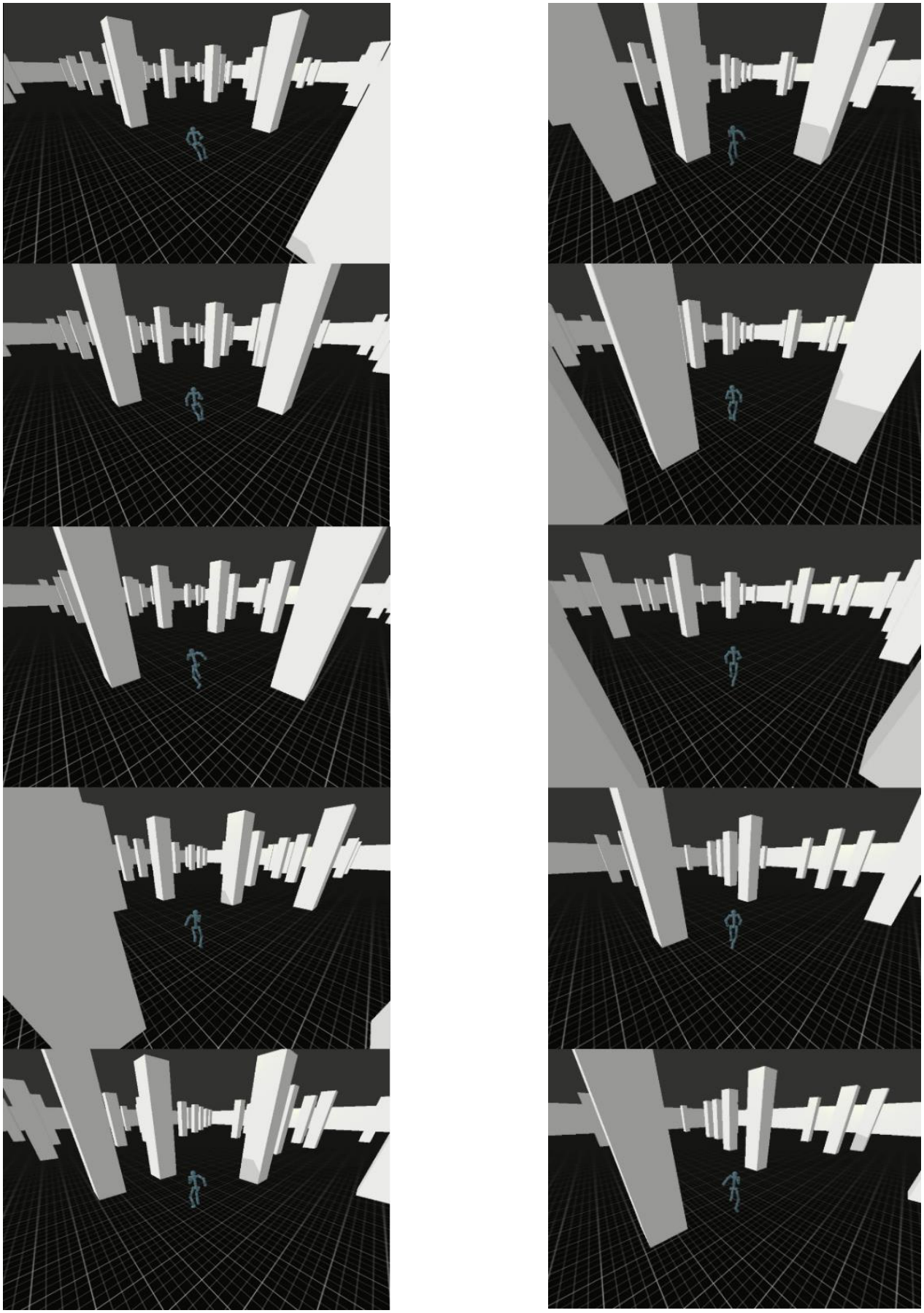


Figure 7: Character simulation snapshots in moving environment. We took 10 screenshots to better show the movement trajectory of the character and obstacles. The order of screenshots is from top to bottom, left to right.

더 빠른 속도로 움직이는 경향이 있었다. 보통 우리가 원하는 캐릭터 모션은 조깅하면서 장애물을 회피하는 것이다. 하지만 effort penalty 를 추가하지 않으면 캐릭터가 조깅보다 빠른 속도로 뛰면서 장애물을 회피한다. 이 부분은 영상을 통해서 확인할 수 있다.

본 논문에서는 effort penalty 를 사용하거나 제거한 상태로 정책을 학습하여 비교하였다. 학습한 정책을 가지고 정지된 장애물 환경과 이동하는 장애물 환경에서 각각 100 번의 테스트를 진행하였다(극단적인 상황 포함). 테스트의 내용은 총 100 번 실험하는데 100 번의 실험 중에 시작 위치부터 목적지까지 장애물과 충돌 없이 도착할 수 있는 횟수를 통계하는 것이다. 통계 결과는 Figure 9 에서 나타난다. 결과를 보면 EP 항목을 제거하고 학습한 정책은 각 환경에서 훨씬 더 좋은 장애물 회피 성능을 보여주고 있는 것으로 나타난다.

6.4 장애물의 크기와 속도

5.2.1 절에서 설명했듯이 정지된 장애물 환경에서 학습을 진행할 때 장애물의 변의 길이는 4 미터였다. 5.2.2 절에서 설명한 움직이는 장애물 환경에서는 장애물에 $0\sim0.15m/timestep$의 랜덤 속도를 주고 시뮬레이션하였다. 정지된 장애물 환경에서 학습한 정책을 움직이는 장애물 환경에서 적용할 때, 장애물의 유효한 속도나 크기의 범위는 실험을 통해서 결정하였다. 장애물의 크기를 4 로 고정시킨 경우 장애물에 랜덤한 방향으로 $0\sim0.3m/timestep$의 랜덤 속도를 주고 시뮬레이션하면 케릭터는 무난하게 장애물을 피하면서 목적지에 도착할 수 있다. 이 수치를 초과하면 장애물의 속도가 캐릭터의 속도보다 더 빨라 캐릭터가 회피할 수 없는 상황이 발생한다. 장애물의 속도를 랜덤 방향의 $0\sim0.15m/timestep$의 랜덤 속도로 설정한 경우 장애물의 크기는 최대 8 미터까지 견고하게 지원할 수 있다 (첨부 영상 참고).

7. 결론 및 향후 계획

Motion VAE 모델의 강력한 생성 능력 때문에 본 논문에서는 72 시간 안에 정지된 장애물을 피하면서 목적지에 도착할 수 있는 정책을 학습할 수 있었다. 본 연구에서 정지된 장애물 환경에서 학습한 정책을 그대로 움직이는 장애물 환경에 옮겨서 사용할 수 있는 것이 장점이다. 본 연구에서 학습한 모델의 성능은 극단적인 상황(실제 사람이라도 장애물을 회피할 수 없는 상황)이 아닌 이상 90%이상의 성공율로 장애물의 주변에서 경로를 찾아서 통과할 수 있었다.

본 연구의 부족한 점도 존재한다. 장애물 회피 태스크에서 최고의 성능을 달성하기 위해서는 필요할 때 캐릭터가 최대한의 속도로 움직이는 것이 좋다. 그래야 모든 장애물을 자연스럽게 잘 피하면서도 목적지에 빠르게 도착할 수 있을 것이다. 하지만 아직은 장애물을 피하는 성능을 과도하게 추구하면, 캐릭터가 조심스러워 지면서 이동 모션이 느려지거나 부자연스러워질 수 있다. 장애물을 회피해야 할 때 빠르게 움직이고 주변에 장애물이 없을 때는 자연스럽게 뛰며 유연하게 행동하는 캐릭터 에이전트를 학습시키는 것은 본 연구의 향후 연구 목표이다. 향후 이 목표를 달성하기 위하여 계층적 학습 구조를 도입하거나, 보상 함수를 수정하는 등 연구를 계속 진행할 것이다.

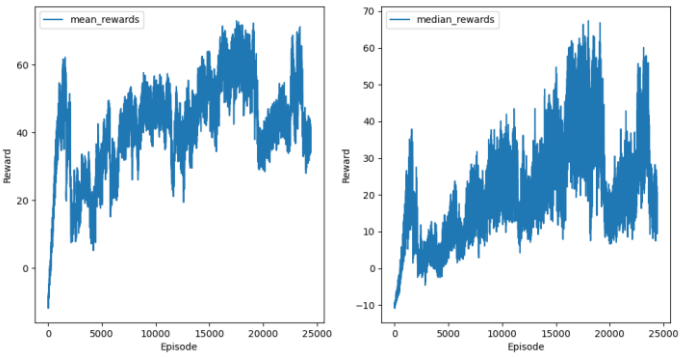


Figure 8: Mean and median value of return. Due to the specificity of the learning task, the resulting reward function curve does not look smooth. We're using 100 agents learning simultaneously, each with a different starting position—some are near obstacles while others are far. This causes significant fluctuations in the mean and median of the reward function. In this task, referring to the mean learning curve is more meaningful as it reflects the gradual improvement in the learned policy.

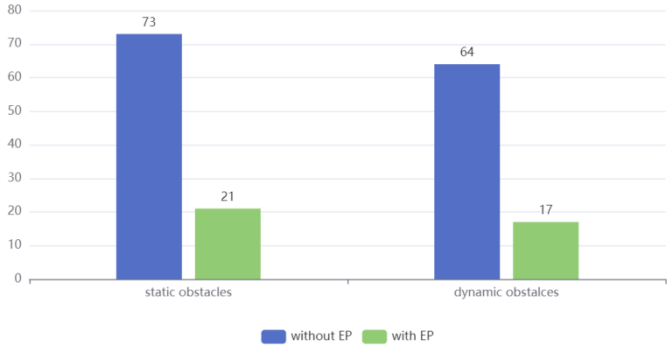


Figure 9: The number of times the destination can be successfully reached in 100 tests

감사의 글

이 논문은 2021 년도 정부(한국전자통신연구원)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No. 2021-0-00320, 실 공간 대상 XR 생성 및 변형/증강 기술 개발

References

[1] Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel van de Panne. 2020. Character Controllers Using Motion VAEs. ACM Trans. Graph. 39, 4, Article 40 (July 2020), 12 pages. https://doi.org/10.1145/3386569.3392422

[2] Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel van de Panne. 2017. DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. ACM Trans. Graph. 36, 4, Article 41 (July 2017), 16 pages. DOI: http://dx.doi.org/10.1145/3072959.3073602

[3] Jaedong Lee, Jungdam Won, and Jehee Lee. 2018. Crowd Simulation by Deep Reinforcement Learning. In Proceedings of Motion, Interaction and Games, Limassol, Cyprus, November 8-10, 2018, 7 pages. DOI: 10.1145/3230744.3230782

[4] V. Lumelsky and T. Skewis, “Incorporating range sensing in the robot navigation function,” IEEE Transactions on Systems Man and Cybernetics, vol. 20, pp. 1058 – 1068, 1990.

[5] V. Lumelsky and Stepanov, “Path-planning srategies for a point mobile automaton amidst unknown obstacles of arbitrary shape,” in Autonomous Robots Vehicles, I.J. Cox, G.T. Wilfong (Eds), New York, Springer, pp. 1058 – 1068, 1990.

[6] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” International Journal of Robotics Research, vol. 5, no. 1, pp. 90–98, 1995.

[7] J. Borenstein and Y. Koren, “The vector field histogram - fast obstacle avoidance for mobile robots,” IEEE Transaction on Robotics and Automation, vol. 7, no. 3, pp. 278 – 288, 1991.

[8]Volodymyr Mnih, Koray Kavukcuoglu. 2015. Human-level control through deep reinforcement learning. Nature 518 (2015), 529–533.

[9] David Silver, Aja Huang. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. Nature 529, 7587 (2016), 484–489. TensorFlow. 2015. TensorFlow: Large-Scale Machine L

[10] LEE Y., WAMPLER K., BERNSTEIN G., POPOVIC´ J., POPOVIC´ Z.: Motion fields for interactive character locomotion. In ACM SIGGRAPH Asia 2010 papers. 2010, pp. 1–8.

[11] LEVINE S., WANG J. M., HARAUX A., POPOVIC´ Z., KOLTUN V.: Continuous character control with low-dimensional embeddings. ACM Transactions on Graphics (TOG) 31, 4 (2012), 1

[12] COROS S., BEAUDOIN P., VAN DE PANNE M.: Robust task based control policies for physics-based characters. In ACM SIGGRAPH Asia 2009 papers. 2009, pp. 1–9.

[13] PENG X. B., BERSETH G., VAN DE PANNE M.: Dynamic terrain traversal skills using reinforcement learning. ACM Transactions on Graphics (TOG) 34, 4 (2015), 1–11.

[14] BROCKMAN G., CHEUNG V., PETTERSSON L., SCHNEIDER J., SCHULMAN J., TANG J., ZAREMBA W.: Openai gym. arXiv preprint arXiv:1606.01540 (2016).

[15] DUAN Y., CHEN X., HOUTHOOFT R., SCHULMAN J., ABBEEL P.: Benchmarking deep reinforcement learning for continuous control. In International conference on machine learning (2016), PMLR, pp. 1329–1338.

[16] LEE S., PARK M., LEE K., LEE J.: Scalable muscle-actuated human simulation and control. ACM Transactions On Graphics (TOG) 38, 4 (2019), 1–13.

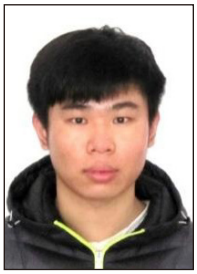
[17] Ilya Kostrikov. 2018. PyTorch Implementations of Reinforcement Learning Algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail.

[18] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. 2018. DeepMimic: Example-Guided Deep Reinforcement Learning of Physics Based Character Skills. ACM Trans. Graph. 37, 4, Article 143 (August 2018), 18 pages. https://doi.org/10.1145/3197517.3201311

[19] He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode Adaptive Neural Networks for Quadruped Motion Control. ACM Trans. Graph. 37, 4, Article 145 (August 2018), 11 pages. https://doi.org/10.1145/ 3197517.3201366

[20] Taesoo Kwon, Taehong Gu, Jaewon Ahn, and Yoonsang Lee. 2023. Adaptive Tracking of a Single-Rigid-Body Character in Various Environments. In SIGGRAPH Asia 2023 Conference Papers (SA Conference Papers ’23), December 12–15, 2023, Sydney, NSW, Australia. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3610548.3618187

〈저자소개〉

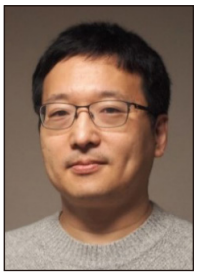


사 정

- 2017-2021 한양대학교 컴퓨터소프트웨어학부 학사
- 2021-2024 한양대학교 컴퓨터소프트웨어학과 석사
- 관심 분야: Reinforcement learning, Physically-Based Character Control
- https://orcid.org/0009-0009-0545-4187

구 태 홍

- 2006-2015 한양대학교 컴퓨터공학부 학사
- 2015-2018 한양대학교 지능형로봇학과 석사
- 2018-2024 한양대학교 컴퓨터소프트웨어학과 박사
- 관심 분야: Physics-based models, Machine learning techniques.
- https://orcid.org/0000-0003-0590-153X



홍 길 동

- 1996-2000 서울대학교 전기컴퓨터공학부 학사
- 2000-2002 서울대학교 전기컴퓨터공학부 석사
- 2002-2007 한국과학기술원 전산학전공 박사
- 관심 분야: Physics-based models, Machine learning techniques.
- https://orcid.org/0000-0002-9253-2156