공학석사학위논문

다양한 동작을 수행할 수 있는 물리 기반 캐릭터 컨트롤러

Physics-based Character Controller with Diverse Motor Skills

2022년 8월

서울대학교 대학원

컴퓨터공학부

이 용 우

다양한 동작을 수행할 수 있는 물리 기반 캐릭터 컨트롤러

Physics-based Character Controller with Diverse Motor Skills

지도교수 서 진 욱

이 논문을 공학석사 학위논문으로 제출함

2022년 8월

서울대학교 대학원

컴퓨터공학부

이 용 우

이 용 우의 공학석사 학위 논문을 인준함

2022년 8월

위 원 장: 김 명 수 (인)

부위원장: 서 진 욱 (인)

위 원: 신 영 길 (인)

요 약

본 논문의 목표는 물리 시뮬레이션 상에서 캐릭터가 여러 가지 액션을 수행할 능력을 갖춘 뒤 유저의 지시에 맞게 동작을 하는 것이다. 유저는 캐릭터에게 명령을 내리면 캐릭터는 현재 자세에서 해당 동작을 수행하기 위해 계획적으로 동작한다. 우리는 별개의 모션 클립 데이터들을 통해 캐릭터가 학습할 수 있는 모션의 가짓수를 최대한 다양화하고 각 모션들을 높은 퀄리티로 학습시킬 수 있는 프레임 워크를 소개한다. 물리 기반 캐릭터를 컨트롤하기 위한 전체적인 네트워크는 캐릭터의 행동을 생성하는 심층강화학습(Deep-RL)기반 컨트롤러와 컨트롤러가 알맞은 동작을 생성할 수 있도록 유도하는 Discriminator로 이루어져 있다. Discriminator는 Deep-RL 컨트롤러에서 나온 결과를 가지고 학습하고 Deep-RL 컨트롤러는 Discriminator에서 나온 결과를 가지고 학습함으로써 두 네트워크가 서로 적대적으로 학습한다. 본 논문은 이러한 접근방식이 효과적임을 입증하기 위해 하나의 캐릭터가 확연히 다른 역동적인 움직임들을 유저의 입력에 맞게 순차적으로 행할 수 있는 유저 상호작용이 가능한 예시를 보여준다.

주요어: 컴퓨터그래픽스, 캐릭터 애니메이션, 물리 시뮬레이션, 강화학습

학 번: 2020-22001

차 례

표 차례

그림 차례

제 1 장 서론

물리 시뮬레이션 속에서의 캐릭터 컨트롤은 컴퓨터 그래픽스 분야에서 오랫동안 연구되어온 주제이다. 물리 시뮬레이션의 캐릭터는 주변 환경과 상호작용을 할 수도 있고 물리적으로 자연스러운 모션이 나타나는 장점이 있다. 그러나 물리적인 조건들을 만족하면서 캐릭터가 넘어지지 않고 사람처럼 움직이게 제어하는 것은 컴퓨터 그래픽스와 로보틱스 두 분야 모두에서 해결하기 어려운 주제였다. 시뮬레이션 속 캐릭터를 사람처럼 걷게하기 위해서 많은 연구자들이 사람의 모션캡쳐 데이터를 사용한 data-driven 방식으로 캐릭터가 넘어지지 않고 이족보행을 할수 있도록 시도했다. 사람의 모션을 그대로 따라하도록 제어한 뒤 캐릭터의 밸런스를 피드백 제어를 통해 보정했다.
최근에는 심층 강화학습의 발전으로 이족보행을 넘어 캐릭터가 더욱 역동적인 모션을 수행할 수 있게 되었다. 모션캡쳐 데이터가 주어지면 해당 모션을 최대한 유사하게 따라할 수 있도록 강화학습이 진행됐다. 하지만 이 방식은 캐릭터가 따라해야할 모션 시퀀스가 정해져야 했다. 따라서 유저가 캐릭터 모션에 개입하기 어려웠고, 유저가 캐릭터를 제어하려면 별도의 모션 컨트롤러가 필요했다. 별도의 모션 컨트롤러가 유저의 명령에 맞게 캐릭터가 따라할 모션을 생성하면 캐릭터가 그 모션을 따라가는 방식으로 연구가 진행되었다.

본 논문은 별도의 모션 컨트롤러를 만들지 않고 캐릭터가 유저의 명령에 따라 다양한 동작을 수행할 수 있는 시스템을 제시한다. 따라해야할 별도의 모션 시퀀스를 만들지 않고 캐릭터가 유저의 명령을 듣고 스스로 적절한 모션을 생성한다. 따라해야할 모션 시퀀스가 없기 때문에, 캐릭터는 같은 포즈에서도 유저 명령에 따라 다양한 동작들을 수행할 수 있다. 캐릭터는 우리의 control policy에 의해 제어되는데, control policy는 여러 가지 동작들을 포함한 모션캡쳐 데이터를 학습하고 캐릭터의

현재 상태와 유저의 명령에 맞는 적절한 모션들을 생성한다.

우리 시스템은 control policy가 모션캡쳐 데이터로부터 여러가지 동작을 학습하는 데 있어 적대적 학습을 사용했다. 따라서 강화학습에 필요한 reward 함수들을 discriminator에서 나온 값을 가지고 정의한다. 이전의 적대적 학습 기반 캐릭터 컨트롤과 우리 시스템의 차별점은 적대적 학습을 하는 과정에서 conditional discriminator를 사용한다는 점이다. 우리의 conditional discriminator는 캐릭터의 모션과 모션캡쳐 데이터를 구분할 수 있을 뿐만 아니라 주어진 모션이 어떤 유형의 동작인지 분류할 수 있다. 이를 통해 discriminator의 분류 성능을 향상시켰다.

우리는 적대적 학습방식에 컨디션을 부여함으로써 다양한 이점들을 얻을 수 있었다. 먼저 우리의 시스템은 적대적학습을 사용한 이전 연구들보다 더욱 다양한 동작들을 캐릭터에게 학습시킬 수 있었다. discriminator의 성능을 향상되어 학습할 수 있는 모션의 폭이 넓어졌다. 두 번째로 캐릭터 컨트롤러를 강화학습의 reward 함수를 기반으로 만들 수 있다. 우리는 이 reward 함수를 control reward라고 정의했고 유저의 명령이 새로 주어지면 캐릭터는 control reward를 통해 유저의 명령대로 움직이도록 학습한다. 마지막으로 우리는 유저가 명령을 바꿔 캐릭터가 동작을 전환할 때 전환동작이 모션캡쳐 데이터로 존재하지 않더라도 스스로 적절한 동작을 생성할 수 있다. 두 동작 사이의 모션을 취하도록 control reward를 적절히 조절하면 유저가 전환 동작을 직접 만들지 않아도 캐릭터가 자연스러운 모션을 스스로 탐색한다.

제 2 장 관련 연구

2.1 물리 시뮬레이션 기반 캐릭터 제어

컴퓨터 속 캐릭터가 사람같은 동작을 하도록 제어하는 것은 컴퓨터 그래픽스 분야에서 오랫동안 연구되어온 주제이다. Hodgins는 Finite state machine과 신체 밸런스를 유지하기 위한 별도의 알고리즘으로 가상의 캐릭터가 역동적인 동작, 예를 들어 달리기, 점프, 자전거 타기 등을 할수 있는 컨트롤러를 제안했다 [1]. Hodgins와 Pollard는 이전에 발표한 컨트롤러가 크기가 다양한 여러 가지 캐릭터들도 컨트롤이 가능하도록 최적화하기도 했다 [2]. 키네마틱 제어를 넘어 물리적인 환경 속에서 캐릭터를 제어하는 건 기존의 캐릭터 제어보다 어려운 주제이다. 그러나 물리 시뮬레이션 기반 캐릭터 제어는 물리적으로 가능한 움직임을 만드는 점에 있어 키네마틱 제어에 비해 이점이 있다. 다양한 동작을 재현하기 전에 두 발로 물리 환경 상에서 중심을 잡을 수 있다는 것 자체가 힘든 일이기 때문이다. 이족 보행을 성공적으로 제어하기 위한 연구는 컴퓨터 그래픽스와 로보틱스가 함께 연구해온 주제이다. 이후에도 다양한 물리 시뮬레이션 상 캐릭터 컨트롤러가 제안되었으나, 그 중에서도 Yin의 SIMBICON이 robust하면서 주어진 레퍼런스를 잘 따라갈 수 있는 피드백 기반 보행 컨트롤러로 인정받고 있다 [3]. SIMBICON은 기존의 애니메이터들이 사용했던 Finite state machine을 기반으로 캐릭터가 움직일 포즈를 정해준 뒤 움직이면서도 균형이나 그 외 물리적인 조건들을 위해 피드백 컨트롤 시스템을 별도로 만들었다. 그 외에도 균형잡힌 이족보행을 위해 Inverted pendulum 모델의 아이디어를 적용하기도 했다[4].

균형잡힌 캐릭터의 안정적인 움직임을 위해 optimization을 활용한 연구도 많이 제안되어왔다. Wang et al.은 생체역학적인 측면을 고려해 SIMBICON이 더 사람처럼 걷도록 최적화를 했다 [5]. 많은 연구들은 캐릭터가 움직이는데 있어 에너지 소비를

최소화하는 방향으로 trajectory optimization을 해 컨트롤러를 제안했다 [6, 7, 8, 9].

모션 캡쳐 데이터를 활용한 데이터 기반 캐릭터 제어는 캐릭터의 움직임을 좀 더 사람에 가깝게 만들었다. 하지만 모션 캡쳐 데이터를 시뮬레이션 상에 복원하면 물리적으로 부정확한 경우가 많다. 따라서 데이터 기반으로 캐릭터를 제어하면 밸런스를 유지하기 어렵기 때문에 일부 연구는 상체만 모션캡처 데이터를 사용하고 하체는 기존의 컨트롤러를 사용하기도 했다 [10, 11]. [12]은 모션캡처 데이터의 물리적인 오차를 spacetime optimization을 통해 교정하고 이차원상의 이족 보행을 데이터 기반으로 제어했다. [13]은 short-horizon tracking과 quadratic programming을 추가해 이족보행의 밸런스를 향상시켜 3차원 상에서 단순화된 3-link 캐릭터의 제어를 보였다. [14]은 3차원에서 사람 전체의 DOF를 컨트롤하는 것을 Non-linear Quadratic Regulator를 가지고 시도했다. [15]은 별도의 optimization 없이 모션캡처 데이터에 밸런스를 위한 피드백을 추가하여 사람의 모든 DOF를 활용해 3차원 상에서 캐릭터의 보행 제어를 했다.

2.2 심층 강화학습을 통한 물리 기반 캐릭터 제어

심층 강화학습을 기반으로 물리 시뮬레이션에서 캐릭터를 제어하면 기존의 최적화기법들보다 짧은 시간에 더 나은 퀄리티로 캐릭터를 제어할 수 있다. 특히 모션 데이터를 레퍼런스로 사용해 캐릭터가 매 순간 모션캡쳐 데이터와 유사하도록 학습하는 Imitation learning은 캐릭터의 동작이 이전 방식보다 사실적으로 개선됨과 동시에 시뮬레이션 상에서 밸런스를 유지할 수 있었다 [16, 17]. Imitation learning은 단일 모션 클립을 학습할 때는 매 순간 타겟 포즈가 정해져 있지만 여러 개의 다른 모션들을 포함한 데이터셋을 학습할 때는 해당 순간에 따라해야할 타겟 포즈를 정하기 어렵다. 최근 방법들은 다양한 모션들로 확장하기 위해 별도의 컨트롤러를 만들어

컨트롤러가 데이터셋을 학습했다. 모션 컨트롤러가 유저의 명령에 맞는 최적의 타겟 포즈를 정해주고 캐릭터가 타겟 포즈를 따라하도록 학습해 다양한 모션들을 수행할 수 있었다 [18, 19, 20, 21]. 따라서 다양한 모션 데이터셋을 학습하는 별도의 High-level 키네마틱 컨트롤러를 필요로 한다 [18, 20, 22]. 그러나 적합한 타겟 포즈를 찾아주는 효과적인 키네마틱 컨트롤러를 설계하는 일은 강화학습과 별개로 많은 수작업과 섬세한 환경 설정이 필요하다. 이번 연구에서 우리는 별도의 컨트롤러 없이 유저의 입력에 따라 다양한 동작을 구사하는 캐릭터 컨트롤러를 구현하는 것이 목적이다. 이를 목적으로 우리는 Generative Adversarial Imitation Learning(GAIL) [23]을 메인 알고리즘으로 한 시스템을 제안한다.

2.3 적대적 모방학습

Generative Adversarial Imitation Learning(GAIL)은 Generative Adversarial Network (GAN) [24]의 근본적인 아이디어를 Imitation learning에 적용한 모델이다. GAN과 마찬가지로 GAIL은 2개의 네트워크가 서로 적대적으로 대립하며 min-max 게임을 하지만 강화학습의 policy가 생성자역할을 한다는 면에서 GAN과 다르다. Discriminator는 Agent의 행동을 Expert의 행동으로부터 구분하도록 학습하고 Policy는 Expert를 모방해 Discriminator를 속일 수 있도록 학습한다. Discriminator는 Policy의 학습을 위한 일종의 Objective function의 역할을 한다. Ho et al. [23]의 연구에 따르면 에피소드가 진행될 동안 Agent의 상태와 Policy의 액션을 쌍으로 (s_t, a_t)를 수집한다. 그들의 연구에 따르면 Discriminator가 속을 수 있는 (s_t, a_t)을 만들었을 때 높은 보상을 받는다. 한편 적대적 학습 방식들의 근본적인 문제점은 내쉬 균형을 찾는 것이 어려운 문제이기 때문에 학습이 불안정하다. Merel et al. [25]은 discriminator의 학습을 위한 Observation으로 (s_t, a_t) 대신 (s_t, s_{t+1})를 사용해 Agent의 상태 정보만을 가지고 imitation learning을 시도했다. Peng et al. [26]은 least square loss

와 gradient penalty를 활용해 GAIL의 안정성을 개선시켰다. 그들의 연구에서 policy는 Goal-conditioned reinforcement learning을 통해 분류되지 않은 데이터셋로부터 Motion prior를 배웠고 특정 임무를 수행하기 위한 동작들을 스스로 만들어 냈다. 캐릭터 컨트롤러의 관점에서 Peng et al. [26]의 Goal-conditioned reinforcement learning은 User가 원하는 동작이 아니라 Goal을 달성하기 쉬운 모션만 나타나는 Mode collapse가 나타날 수 있다. Xu and Karamouzas [27]는 Mode collapse에 빠질 확률을 줄이고자 다수의 Discriminator를 사용했고 다수의 동작 각각의 퀄리티를 위해 다수의 Policy들을 학습한 뒤 번갈아가며 사용했다. 우리는 모션의 종류별 특성 및 데이터 비율과 관계없이 Mode collapse를 예방하고 다수의 Discriminator 쓰지 않고 단일 Discriminator만으로 더 다양한 모션을 고르게 학습하기 위해 GAIL에 추가적인 조건을 부여한 conditional GAIL을 시스템에 적용했다.

2.4 조건부 적대적 생성 모델

조건부 적대적 생성모델(Conditional Generative Adversarial Learning, cGAN)은 데이터를 생성하는 시스템에 추가적인 클래스 정보가 주어졌을 때 클래스에 맞는 결과를 생성하도록 유도하는 모델이다. cGAN은 이미지 생성 분야에서 활발하게 연구된다. Mirza et al. [28]은 기존 GAN에 클래스 정보를 추가함으로써 cGAN을 처음 제안했다. Odena et al. [29]은 Discriminator에 Auxiliary classifier를 추가함으로써 클래스 수를 더 늘려도 퀄리티를 유지할 수 있었다. 그 이후에도 Discriminator의 확장성을 향상하기 위한 많은 테크닉들이 제안되어왔다 [30, 31, 32]. cGAN은 종류에 따른 데이터양이 불균형적일 때도 모든 유형을 균형적으로 학습할 수 있는 장점이 있다. 우리는 이런 cGAN의 장점을 활용한 조건부 적대적 모방학습(Conditional Generative Adversarial Imitation Learning)으로 광범위한 동작들을 수행하는 캐릭터 컨트롤러를 제안한다. 우리는 [29]의 네트워크 모델을 따라 Discriminator가 캐릭터가 만든 동작을 레퍼런스로부터 구분할 뿐만아니라 동작의 클래스도 분류함으로써

단일 Discriminator의 확장성을 개선시켰다. 클래스 예측 결과를 Policy의 Objective function에 포함시켰을 때 Policy는 클래스에 해당하는 동작을 취하고 있을 때 더 많은 보상을 받을 수 있다. 따라서 Policy가 다양한 모션들을 균형적으로 학습할 수 있도록 유도함으로써 모션의 다양성을 늘릴 수 있다.

제 3 장 개 요

우리가 만든 시스템의 목적은 물리 시뮬레이션 속 캐릭터가 유저의 명령에 맞게 다양한 모션들을 수행하는 것이다. 학습한 결과물의 목표는 그림 3.1처럼 시뮬레이션 속 캐릭터에게 유저가 원하는 액션을 명령하면 해당 액션을 수행하는 컨트롤러를 만드는 것이다. 또한 캐릭터가 수행할 수 있는 액션의 종류를 원하는 만큼 종류의 갯수를 N개로 늘릴 수 있어야 한다. 이를 구현하기 위한 우리 시스템의 전반적인 개념도는 그림 3.2과 같다. 시뮬레이션 속 캐릭터는 control policy π가 만들어 주는 모션을 따라간다. 시뮬레이션을 진행하면서 얻은 캐릭터의 모션들은 discriminator의 학습을 위한 데이터로 저장된다. discriminator는 시뮬레이션 캐릭터의 모션 데이터와 레퍼런스로 제공되는 모션캡처 데이터를 동시에 입력받고, 임의의 데이터를 받았을 때 그 데이터가 모션캡처 데이터인지 시뮬레이션 데이터인지 구분한다. 또한 discriminator는 해당 모션 데이터가 어떤 모션 클래스에 속하는지 분류를 한다. Discriminator는 임의의 데이터에 대해 시뮬레이션 데이터인지 모션캡처 데이터인지 잘 구분하도록 학습함과 동시에 해당 모션이 어떤 유저의 명령의 의해 만들어진 건지 예측하도록 학습한다. 한편 캐릭터의 모션을 만들어주는 Control policy π는 Discriminator를 속여 모션캡처 데이터로 인식하도록 학습함과 동시에 유저의 명령이 주어졌을 때 명령에 맞는 모션을 취하도록 학습한다. 이러한 학습의 형태는 Policy π와 Discriminator가 적대적으로 학습하는 형태로, 적대적 모방학습(GAIL) 알고리즘을 기반으로 하고 있다. 특히 우리는 조건부 적대적 모방학습(conditional GAIL, cGAIL)을 활용했고 이에 대한 내용을 이번 챕터에 다룰 예정이다. 더불어 이번 챕터는 본 논문이 수행하는 물리 시뮬레이션에서의 캐릭터 제어를 위한 이론적 배경들을 소개한다.

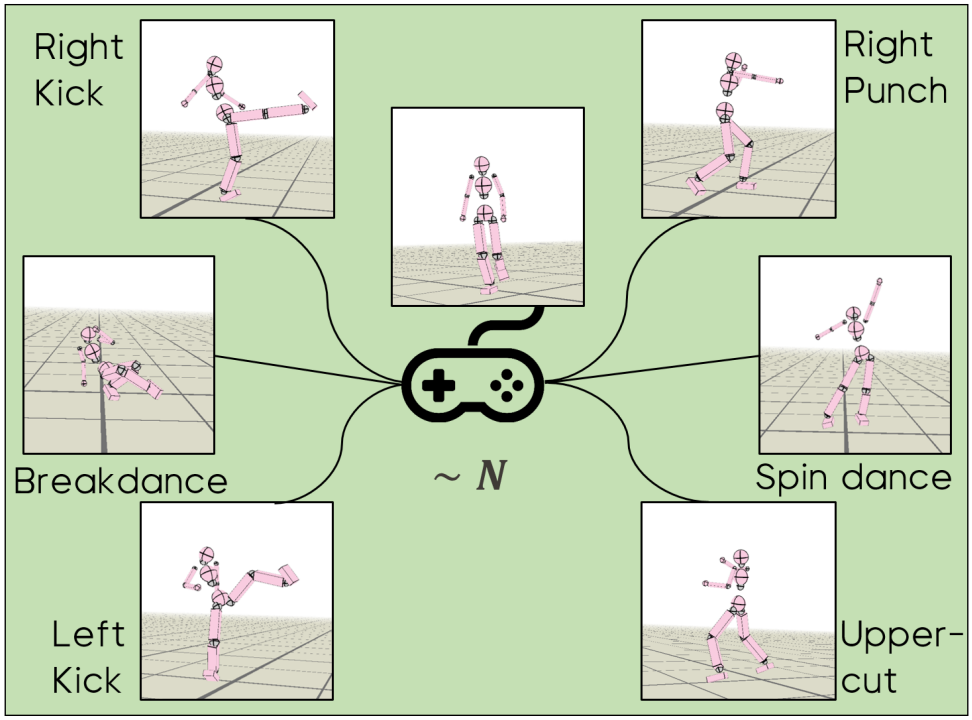

그림 3.1: 구현한 시스템의 결과물 예상도

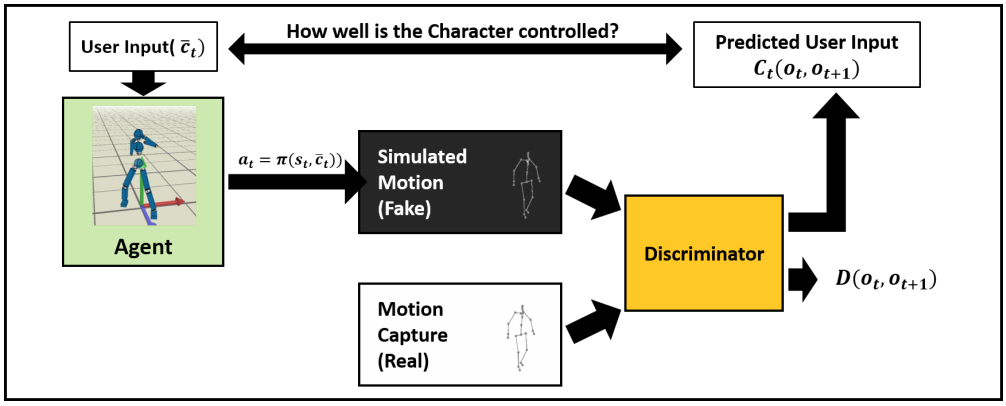

그림 3.2: 시스템의 전반적인 개념도

3.1 이론적 배경

3.1.1 물리 기반 캐릭터 제어

물리 시뮬레이션 속에서의 캐릭터 제어는 캐릭터의 모션이 물리적으로 자연스러운 결과를 만드는데 이점이 있다. 물리적인 제약조건 속에서 캐릭터를 자유롭게 제어하는 연구는 다양한 방식이 제안되어 왔지만 본 논문은 모션 데이터를 기반으로 캐릭터가 매 순간 주어지는 타겟 모션을 따라하는 접근법에 기반을 두고 있다 [33]. 타겟 모션이 주어졌을 때 캐릭터가 이를 빠르게 따라가기 위한 방법은 Proportional Derivative(PD) 제어로 캐릭터의 각 조인트가 타겟 위치로 잘 따라가도록 토크를 가하는 방식을 사용한다 [34]. 일반적인 PD 제어도 타겟 모션을 따라가면서 캐릭터가 안정적으로 균형을 잡을 수 있게 하지만 Stable PD 제어는 내연적 적분법(Implicit Integration)을 통해 기존 PD제어보다 더욱 안정적인 제어를 보여주었다 [35]. 본 논문은 다관절의 강체로 이루어진 캐릭터를 제어하기 위해 Stable PD 제어를 사용해 매 순간마다 주어진 타겟 포즈를 따라하도록 설계했다. 타겟 포즈가 주어져 각 조인트마다 가할 토크는 다음과 같이 계산된다.

$$\tau^n = -k_p(q^n + \Delta t - \bar{q}^{n+1}) - k_d(\dot{q}^n + \Delta t\ddot{q}^n) \tag{3.1}$$

τ^n은 n번째 프레임에서 가할 토크의 크기이고, $q^n, \dot{q}^n, \ddot{q}^n$은 캐릭터 각 조인트의 현재 위치, 속력, 가속력을 나타낸다. $\bar{q}^{n+1}$은 캐릭터가 따라할 조인트의 타겟 위치이다. k_p, k_d는 각각 stiffness 계수와 damping 계수이다.

3.1.2 심층 강화 학습

우리의 캐릭터는 심층 강화학습을 기반으로 매 순간 최적의 모션을 만들어내는 control policy π에 의해 움직인다. 유저의 입장에서 캐릭터는 매 프레임 특정 모션을 수행하라는 control vector $\bar{c}_t$를 받는다. 매 프레임 t마다 캐릭터는 현재 상태 및 여러

가지 정보를 수집한 state s_t를 만들고 s_t와 $\bar{c}_t$를 policy π에게 보내면 π는 그에 맞는 action a_t $\pi(a_t|s_t, \bar{c}_t)$를 생성한다. 본 논문에서 a_t는 캐릭터가 따라할 다음 프레임의 타겟 포즈를 의미한다. 캐릭터는 a_t를 참고해 움직여 다음 상태 s_{t+1}를 생성하고 그에 따른 reward $r_t = r(s_t, a_t, s_{t+1}, \bar{c}_t)$을 받는다. 캐릭터의 목표는 매 프레임마다 r_t를 받아 최종적으로 가장 높은 Expected discounted return $J(\pi)$를 받는 policy를 학습하는 것이다. $J(\pi)$는 다음과 같다.

$$J(\pi) = \mathbb{E}_{p(c)}\mathbb{E}_{p(\tau|\pi,\bar{c})}[\sum_{t=0}^{T-1} \gamma^t r_t], \tag{3.2}$$

이 수식에서 $p(\tau|\pi, \bar{c}) = p(s_0)\prod_{t=0}^{T-1} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t, \bar{c})$는 control policy π가 control vector $\bar{c}$에 따라 모션의 경로 $\tau = (s_t, a_t, r_t)_{t=0}^{T-1}, s_T$가 만들어진 가능성(Likelihood)을 뜻한다. $p(s_0)$은 초기 state에 대한 확률 분포이고 T는 모션 경로가 지속되는 총 시간, $\gamma \in [0, 1)$은 discount factor이다.

3.1.3 적대적 모방 학습

적대적 모방 학습(GAIL)은 기존의 모방학습보다 (Behavioral cloning)보다 많은 양의 데이터(Demonstrations)를 다룰 수 있다는 점에서 장점이 있고, policy와 주어진 데이터에 유사도를 측정해 목적 함수를 설계한다. Ho and Ermon [23]은 GAIL의 Discriminator 학습을 위해 Expert의 상태와 액션을 필요로 했다. 그런데 모션 캡처 데이터가 Expert의 역할을 하면 Expert의 액션은 알 수 없고 상태만 데이터에 존재한다. Torabi et al. [36]은 GAIL을 Expert의 상태값만 가지고 학습하고자 Discriminator에게 상태-액션 $D(s, a)$ 대신 상태 변화$D(s, s')$를 입력으로 주었다. 본 논문은 Torabi et al.의 방법을 따라 Discriminator $D(s_t, s_{t+1})$이 사용되는데, $D(s_t, s_{t+1})$는 주어진 (s_t, s_{t+1})이 제공된 모션캡처 데이터에서 생성된 건지 policy π에 의해 생성된 건지

를 예측하도록 학습한다. 이에 대한 목적함수는 아래와 같다.

$$\min_{D} -\mathbb{E}_{d^{\mathcal{M}}(s,s')}[\log(D(s,s'))] - \mathbb{E}_{d^{\pi}(s,s')}[\log(1-D(s,s'))]. \tag{3.3}$$

$d^{\mathcal{M}}(s,s')$와 $d^{\pi}(s,s')$는 각각 Expert역할을 할 모션 캡처 데이터셋 $\mathcal{M}$에서와 Policy π에서 s에서 s'으로 캐릭터 상태가 변할 때 확률 분포이다.

한편 policy π는 Discriminator를 속일 수 있을 정도로 모션캡처 데이터처럼 사실적인 캐릭터의 모션을 만들기 위해 아래와 같은 방식의 reward function을 설계해 학습할 수 있다. $r_t = -\log(1-D(s_t,a_t)$ 이러한 적대적 학습 방식은 목적함수가 $d^{\mathcal{M}}(s,s')$와 $d^{\pi}(s,s')$간의 Jensen-Shannon divergence를 줄이는 방향으로 최적화되어 가는 과정이라고 해석할 수 있다 [37].

제 4 장 cGAIL 기반 캐릭터 컨트롤러

먼저 우리는 캐릭터의 모션을 생성하는 control policy π가 다양한 모션들을 학습할 수 있는 알고리즘을 제안한다. 그 뒤 유저가 명령한 모션을 수행하는 알고리즘은 다음 챕터에서 후술할 예정이다. 우리는 policy π가 다양한 모션을 학습했을 때 높은 reward를 받을 수 있도록 아래와 같이 reward 함수를 설계했다.

$$r_t(s_t, a_t, s_{t+1}, \bar{c}_t, g_t) = w^S r^S(s_t, s_{t+1}) + w^C r^C(s_t, s_{t+1}, \bar{c}_t) + w^G r^G(s_t, a_t, s_{t+1}, g_t) \tag{4.1}$$

s_t, s_{t+1}은 각각 시간 $t, t+1$일때 캐릭터의 상태를 뜻하고, a_t는 Policy의 결과로 캐릭터가 취한 액션을 뜻한다. $\bar{c}_t$은 Control Vector이고 유저가 원하는 타겟 모션 타입이 뭔지 나타낸다. g_t는 유저가 원하는 임무에 대한 Goal이다. 위의 세 보상함수는 선형적으로 각각 w^S, w^C, w^G가 곱해져 최종 보상함수 r_t가 계산된다. 위의 reward 함수가 의미하는 바는 policy π가 1) 캐릭터가 사실적인 모션을 수행하는 지(r^S), 2) 유저가 원하는 종류의 모션을 수행하고 있는지(r^C), 3) 유저의 부가적인 과제를 잘 수행하고 있는지(r^G)를 의미하고 policy π는 이 3가지 임무를 잘 수행할 수있는 방향으로 학습해나간다.

우리는 r^S와 r^C, 그리고 r^G의 계산식을 만듬에 있어 GAIL을 활용하려고 한다. 더불어 우리는 기존의 GAIL의 Discriminator부분에 condition을 더한 conditional GAIL을 활용해 reward term들을 설계했다. conditional GAIL을 구성하는 Discriminator와 Policy가 각각 어떻게 학습하는지, 그 결과 캐릭터가 어떻게 다양한 모션들을 수행할 수 있는지에 대해 설명한다.

4.1 다양한 종류의 모션 학습

Conditional GAN은 모델에 추가적인 정보를 제공함으로써 유저가 원하는 데이터를 만들어내도록 유도할 수 있다. 또한 Discriminator가 데이터의 출처를 구분할 뿐만아니라 모션의 타입까지 분류해야 하기 때문에 더 많은 학습을 요구하고 높은 성능을 기대할 수 있다. 우리는 유저가 원하는 모션을 캐릭터가 수행하기 위해 Discriminator에 Condition을 부여했다. Discriminator의 학습을 위한 목적함수는 아래와 같다.

$$\underset{D}{\operatorname{argmin}} L_{style} + L_{class} + L_{gp} \tag{4.2}$$

L_{style}과 L_{class}, L_{gp}는 각각 style loss와 classification loss, 그리고 gradient penalty loss이다. 이번 챕터는 Discriminator의 학습을 위한 목적함수 설계에 각 요소들이 포함된 배경에 대해 설명한다.

4.1.1 적대적 학습 기반 모션 학습

Torabi et al. [36]는 GAIL을 구현할 때 Agent의 State 정보들만 Discriminator에게 제공해 학습시켰다.

$$\underset{D}{\operatorname{argmin}} -\mathbb{E}_{d^{\mathcal{M}}(s,s')}[\log\left(D(s,s')\right)] - \mathbb{E}_{d^{\pi}(o,o')}[\log\left(1-D(s,s')\right] \tag{4.3}$$

$d^{\mathcal{M}}(s,s')$과 $d^{\pi}(s,s')$는 state가 s에서 s'으로 전환되는 경우가 각각 모션캡쳐 데이터셋 $\mathcal{M}$와 Policy π 에서 생성된 모션들에서 나타날 확률이다. $D(s,s')$은 Discriminator가 주어진 모션에 대해서 진위를 구분한 값으로 $[0,1)$의 값을 가진다. 위와 같은 목적함수를 가지고 Discriminator를 학습시키면 학습 수렴 결과 Discriminator가 임의의 모션에 대해 출처가 모션 캡쳐 데이터셋인지 Policy π인지 구분할 수 있게 된다.

Peng et al. [26]은 least-squares GAN(LSGAN)을 사용해 적대적 학습의 불안정성

을 해결했고 그 결과 캐릭터가 높은 품질의 모션을 표현할 수 있었다 [38]. LSGAN은 $D(s, s')$의 범위가 -1과 1사이로 나타나고 데이터셋에 있던 데이터는 1로, 그리고 생성된 데이터는 -1로 구분하도록 학습한다. 우리는 LSGAN의 학습 안정성이 뛰어난 점을 참고해 Discriminator를 학습할 때 아래와 같은 목적함수 L_{style}를 포함시켰다.

$$L_{style} = \mathbb{E}_{d^{\mathcal{M}}(s,s')}[(D(o, o') - 1)^2] + \mathbb{E}_{d^{\pi}(o,o')}[(D(o, o') + 1)^2] \tag{4.4}$$

또한 Discriminator의 입력을 (s, s')대신 State의 일부만을 포함한 Observation pair (o, o')으로 대체 했다. Discriminator observation에 대해서는 챕터5.2에서 후술한다.

4.1.2 모션 분류기

Odena et al. [29]은 GAN을 만들 때 condition을 부여해 이미지를 생성하기 위해 Discriminator에 auxiliary classifier를 추가한 ACGAN을 제시했다. 그 결과 더욱 다양한 종류의 이미지를 생성할 수 있을 뿐더러 Discriminator의 분류 성능도 향상되었다. 우리의 Discriminator는 ACGAN의 Discriminator 구조를 참고해 임의의 모션의 출처를 구분할 뿐만 아니라 모션의 유형도 구분한다. 이는 Discriminator가 모션에 대해 더 구체적으로 분석해 판단을 내려 더 좋은 성능을 갖추기 위한 목적이다. 또한 Discriminator가 모션을 분류한 결과도 적대적 학습에 이용할 수 있고 이는 챕터 4.2에 언급되어있다. 따라서 우리는 Discriminator를 학습할 때 아래와 같은 목적함수 L_{class}를 포함시켰다.

$$L_{class} = \mathbb{E}_{d^{\mathcal{M}}(s,s')}[\log P(C = \bar{c}|(o, o'))] + \mathbb{E}_{d^{\pi}(o,o')}[\log P(C = \bar{c}|(o, o'))] \tag{4.5}$$

$P(C = \bar{c}|(o, o'))$는 주어진 Observation (o, o')에 대해 Discriminator가 유저 명령 $\bar{c}$대로 인식할 확률이다. $\bar{c}$가 유저의 명령을 표현한 1D 벡터이기 때문에 $P(C = \bar{c}|(o, o'))$ 또한 각각의 모션에 대한 확률을 표현한 1D 벡터이다. L_{class}는 $P(C = \bar{c}|(o, o'))$에 로그함수를 씌운 크로스엔트로피 함수로 만들어져 있다.

4.1.3 적대적 학습의 안정성

Discriminator와 Policy가 서로 적대적으로 학습하는 GAIL은 학습이 수렴하지 않고 불안정한 결과가 발생하기 쉽다는 점이 단점으로 지적되어 왔다. 불안정한 이유 중 하나로는 Discriminator가 레퍼런스가 되는 모션캡쳐 데이터셋에 대해서도 쉽게 수렴하지 않고 큰 radient로 학습을 하는 현상이 있다. 그 결과 Policy 입장에서도 모션캡쳐 데이터셋과 유사한 방향으로 수렴되지 못하고 혼동하여 낮은 품질의 모션의 결과가 나올 수 있다. 따라서 이런 문제를 해결하기 위해 모션캡쳐 데이터셋를 가지고 학습할 때 Discriminator에서 발생하는 Gradient의 크기에 페널티를 주었다 [39]. 다라서 우리는 Discriminator를 학습할 때 아래와 같은 Gradient penalty를 위한 목적함수 L_{gp}를 포함 시켰다.

$$L_{gp} = \frac{w^{gp}}{2} \mathbb{E}_{d^{\mathcal{M}}(s,s')}[\|\nabla_{\phi} D(\phi)|_{\phi=(o,o')}\|^2] \tag{4.6}$$

w_gp는 학습 이전에 정하는 하이퍼파라미터로, 모션의 다양성이 적을 때는 높게 설정했고 다양한 모션이 포함되었거나 역동적인 모션들이 많이 포함되었을 수록 낮은 값을 설정했다. 특별히 언급이 없을 경우 w_gp는 0.5로 설정하고 학습을 수행했다.

4.2 캐릭터 컨트롤러

우리의 캐릭터는 Control policy π가 현재 상황에 맞는 생성한 모션을 따라 움직인다. 따라서 캐릭터의 모션의 품질은 Control policy π가 얼마나 사실적인 모션을 만드는지에 따라 결정된다. 또한 Control policy π는 유저가 원하는 액션 종류에 따라 적절한 모션을 생성해야 한다. 이를 우리는 Reward 함수를 수식 4.1과 같이 설계했다. 이번 챕터는 수식 4.1에 쓰여 있는 Reward 요소 r^S, r^C, 그리고 r^G가 어떻게 구성되어 있는지를 설명한다.

4.2.1 Style Reward 설계

캐릭터가 얼마나 사실적으로 행동하는지 평가하는 것은 기준이 추상적이기 때문에 정의하기 어렵다. 기존의 모방학습 기반 방식들은 매 프레임마다 모션캡쳐 데이터셋으로부터 타겟포즈를 참고해 현재 캐릭터 포즈와 타겟포즈의 차이를 계산했지만, GAIL을 사용할 경우 매 프레임마다 따라해야할 타겟포즈가 정해져 있지 않다. 우리는 Peng et al. [26]의 Adversarial Motion Prior(AMP)의 개념을 참고하여 기존의 모방학습의 Tracking 보상함수와 달리 Discriminator에서 나온 결과물을 가지고 reward 함수를 디자인 했다. 따라서 캐릭터는 단일 모션 클립만 학습하고 매 프레임 수행해야할 타겟 포즈가 정해져 있던 기존 모방학습과 달리 다양한 모션 데이터셋을 학습할 수 있다. 아래 수식은 Discriminator를 기반으로 학습하기 위한 Style reward r^S를 정의한 식이다.

$$r^S(s_t, s_{t+1}) = 1 - \frac{1}{4}(1 - (D(o_t, o_{t+1})))^2 \tag{4.7}$$

모션 정보 o_t는 Discriminator observation으로 캐릭터의 상태 s_t의 일부이고 챕터 5.2에 자세히 다룰예정이다. $D(o_t, o_{t+1})$은 Discriminator가 입력으로 주어진 모션 정보 (o_t, o_{t+1})이 데이터에서 주어진 정보인지 Policy π가 만들어낸 정보인지 구분한 결과로, 캐릭터가 만들어낸 결과면 -1에 가깝게, 주어진 데이터로 인한 정보면 1에 가까운 수치를 출력하도록 설계되어 있다. 따라서 r^S는 [0,1]의 값을 가지게되고 캐릭터가 Discriminator를 속여 데이터로부터 만들어진 정보로 착각하게 만들수록 높은 r^S를 받게 된다. 또한 모션 종류에 따라 reward 함수 구조가 다르지 않기 때문에 모션의 종류에 따라 별도로 섬세한 변수 수정이 필요하지 않다.

4.2.2 Control Reward 설계

Motion planner가 없이 유저의 입력에 맞게 캐릭터를 컨트롤하려면 캐릭터가 유저의 의도를 파악할 수 있어야 한다. 따라서 우리는 Policy에 유저의 타겟모션이 뭔지

나타내는 Control Vector $\bar{c}$를 입력으로 추가했다. 따라서 Policy는 이를 반영한 액션 $a_t = \pi(a_t|s_t, \bar{c_t})$를 출력해 캐릭터를 움직인다. 유저의 명령대로 행동하려면 Policy π가 유저에 의도에 맞는 행동을 했을 때 더 높은 Reward을 받아야 한다. 유저의 명령 $\bar{c}_t$에 대해 Control reward 함수 r^C는 다음과 같이 정의했다.

$$r^C(s_t, s_{t+1}, \bar{c}_t) = 1 - \min(1, \|(C(o_t, o_{t+1}) - \bar{c}_t)\|_1) \tag{4.8}$$

$C(o_t, o_{t+1})$은 Discriminator의 Auxiliary classifier를 통과한 결과 값으로 Discriminator가 유저의 명령 $\bar{c}_t$를 예측한 값이다. r^C는 (0,1]의 범위의 값이고 $C(o_t, o_{t+1})$가 $\bar{c}_t$에 가까울 수록 높은 값을 갖는다. Discriminator가 주어진 모션에 대한 유저의 명령을 잘 판단한다고 가정했을 때 $C(o_t, o_{t+1})$가 $\bar{c}_t$에 가깝다는 것은 현재 캐릭터의 행동이 유저의 명령에 부합했다고 볼 수 있다.

4.2.3 Goal Reward 설계

Goal reward 함수 r^G는 유저의 명령과 별개로 유저가 부가적인 임무를 주었을 때 추가되는 함수이다. 캐릭터가 주어진 임무에 대해 유저가 원하는 Goal에 근접한 행동을 하고 있는 지 알려주는 값이다. 예를 들어 유저가 캐릭터를 원하는 방향으로 이동시키고 싶다면 유저의 명령은 걷기 혹은 뛰기지만 유저가 부여하는 임무는 해당 방향이 된다. 이 연구에서 우리는 Goal reward 함수로 캐릭터가 바라보는 방향 액션을 취하는 방향을 기본적으로 주었고 방향에 대한 reward를 정의한 수식은 아래와 같다.

$$r^G(s_t, s_{t+1}, \bar{d}_t) = 1 - \min(0, (\|(d_t \cdot \bar{d}_t)\|_1)^2) \tag{4.9}$$

d_t와 $\bar{d}_t$는 각각 현재 캐릭터의 이동방향과 유저가 원하는 캐릭터의 이동방향으로 3차원 벡터지만 본 논문에서는 XZ평면 상에서의 방향만 바꿀 수 있도록 정의했다. 또한 부가적인 임무가 없는 모션 유형에 대해서는 Goal reward를 1로 고정했다. Goal

reward 함수를 모션의 유형에 따라 다른 임무가 주어질 수 있고 그에 따라서 다른 형태의 함수가 정의될 수 있다.

4.3 모션 전환

때로는 유저가 캐릭터에게 다른 모션 명령을 내려 $\bar{c}_t$가 바뀔 수 있다. Control reward r^C를 이용해 캐릭터가 유저 명령을 인식하는 것과 별개로 자세가 확연히 다른 두 종류의 액션을 순차적으로 수행하기 위해 두 액션 사이의 중간 모션이 필요하다. 단일 frame만에 $\bar{c}_t$이 갑자기 바뀌면 Policy가 캐릭터의 현재 포즈를 고려하지 않고 다음 액션을 위해 캐릭터의 모션이 급격하게 변할 수 있고 이는 부자연스러운 모션으로 연결된다. 캐릭터가 현재 자세에서 유저 명령에 맞는 모션으로 자연스럽게 전환해야 하기 위해 우리는 Policy가 이를 고려하여 a_t를 출력하도록 유도했다. Policy가 모션이 전환되고 있는 상황을 인식해야하기 때문에 모션이 전환되는 순간에 대해서 $\bar{c}_t$를 새롭게 설계했다. 모션 타입 A에서 모션 타입 B로 전환된다고 가정했을 때 최종적인 $\bar{c}_t$와 r^C의 형태는 다음과 같다.

$$\bar{c}_t, transition = (1 - w_{trans})\bar{c}_A + w_{trans}\bar{c}_B \tag{4.10}$$

$$r^C(o_t, o_{t+1}, \bar{c}_A, \bar{c}_B,) = (1 - w_{trans})r^C(o_t, o_{t+1}, \bar{c}_A) + w_{trans}r^C(o_t, o_{t+1}, \bar{c}_B) \tag{4.11}$$

w_{trans}는 모션이 전환될 때 Transition phase로 사용자의 입력에 의해 변환될 때 [0,1]의 범위 안에서 천천히 증가한다. 완벽히 전환된 시점에서 w_{trans}는 1로 유지된다. w_{trans}가 증가하는 속도에 따라 모션이 변하는 정도가 다르기 때문에 다양한 속도의 Transition phase로 실험을 시행했다. 이와 같이 보상 함수를 설계했을 때 Policy는 모션 타입 A와 모션 타입 B의 중간 모션이 데이터셋에 없더라도 스스로 중간 모션을 적절하게 만들어내어 Motion In-betweening을 할 수 있다.

제 5 장 캐릭터 모델 및 학습 네트워크

우리 시스템의 효과를 평가하기 위해, 우리는 Adversarial Character Controller를 가지고 31개의 자유도를 가진 3D 휴머노이드 캐릭터에게 여러 가지 스킬을 학습시켰다. 캐릭터의 모습과 Configuration은 그림 5.1과 같다. 캐릭터는 총 15개의 신체 부위로 이루어져 있고 각각 조인트를 가지고 있다. 그림 5.1의 캐릭터의 모델에서 파란색은 물리적인 공간을 차지하는 캐릭터의 신체부위를 나타내고 흰색 부분은 각 부위의 조인트를 나타낸다. 루트 조인트를 제외한 모든 조인트는 볼 조인트이고 루트 조인트는 자유롭게 움직여야 하기 때문에 6자유도의 자유 조인트로 설정했다.

이번 챕터에서는 시스템의 여러 구성요소들에 대한 구현 사항을 설명한다. 먼저 강화학습의 구성 요소인 캐릭터의 State와 Action의 정의들을 설명하고, Discriminator에 입력될 Observation을 어떻게 정의했는지를 설명한다. 그리고 학습을 하는 Policy network와 Discriminator network에 대해 설명한다.

5.1 State 및 Action

캐릭터의 상태 s_t는 캐릭터 신체 구조를 묘사하는 특성들로 구성되어있다. 모든 특성들은 캐릭터의 로컬 좌표계를 기준으로 변환된 값들이다. 루트의 위치는 캐릭터의 골반으로 지정했다. 모든 특성들은 캐릭터의 로컬 좌표계를 기준으로 변환된 값들로, 캐릭터의 로컬 좌표계는 z축을 루트가 앞을 향하는 방향으로, y축을 월드 좌표계의 y축으로 지정한뒤 x축을 y축과 z축의 외적한 결과로 설정한 좌표계이다. 특성들에 대한 정보는 아래와 같다. 각 조인트의 회전 각도는 Revolute 조인트일 경우 1자유도, Spherical 조인트일 경우 3자유도를 가지고 값을 주었다.

- 지면으로부터 캐릭터의 루트 높이
- 각 조인트의 오프셋

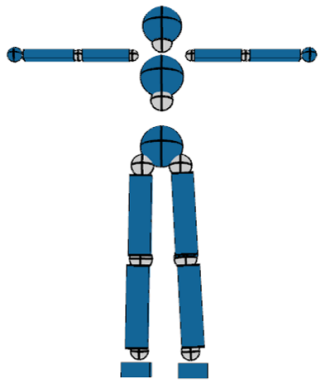

그림 5.1: 사용한 캐릭터 모델

- 각 조인트의 회전 각도
- 각 조인트의 선속도
- 각 조인트의 각속도
- 캐릭터의 무게중심의 3D 위치 및 속도
- 캐릭터 말단 부위의 3D 위치

루트의 위치는 캐릭터의 골반으로 지정했다. 각 조인트의 회전 각도는 조인트의 좌표계의 x축과 y축 벡터의 값을 state에 포함시켰다. 이 모든 특성을 다 고려했을 때 캐릭터의 상태는 총 243차원의 공간에서 정의된다.
캐릭터의 액션 a_t는 캐릭터가 취해야할 포즈를 의미하고 각 조인트의 각도들로 이루어져있다. 각 조인트들의 각도는 3D Exponential Map $q \in \mathbb{R}^3$의 형태로 표현된다. 액션 값은 총 36차원의 공간에서 나타난다.

5.2 Discriminator Observations

Discriminator는 Policy가 모션을 잘 학습하는데 필요한 보상 신호를 주는 역할을 해야 하기 때문에, 현재 캐릭터의 모션을 잘 평가할 수 있는 특성들이 입력으로 들어가야 한다. Discriminator에 입력으로 들어가는 Observation o_t는 캐릭터의 특성을 포함하고 있고 아래와 같다.

- 지면으로부터 캐릭터의 루트 높이
- 루트의 선속도, 각속도
- 각 조인트의 회전 각도
- 각 조인트의 선속도
- 캐릭터 말단 부위의 3D 위치

루트의 위치는 캐릭터의 골반으로 지정했다. 캐릭터의 로컬 좌표계는 State의 기준과 같다. 우리는 Discriminator에게 Observation o_t와 액션 a_t가 가해져 캐릭터 상태가 변한 이후 o_{t+1}를 묶어 2프레임 의 정보인 (o_t, o_{t+1})를 제공한다. 따라서 캐릭터의 상태 변화를 통해 전반적인 캐릭터 모션의 정보를 인식하고 a_t을 직접 명시하지 않아도 Policy가 성공적으로 수행했는지 평가할 수 있다. 또한 o_t에 유저의 입력과 관련된 정보를 넣지 않음으로써 캐릭터의 임의의 포즈가 특정 모션 종류에만 국한되지 않고 여러 모션들에 속해 있을 수 있는 가능성을 고려했다.

5.3 Network Architecture

이 논문에서 사용한 Policy π와 Discriminator의 대략적인 네트워크의 구조는 그림 5.2에 나타나 있다. Policy 네트워크 π와 Value function 네트워크 $V(s_t, \bar{c}_t)$는 둘다 fully-connected 네트워크로 설계되었다. Policy π는 캐릭터의 상태 s_t와 유저 입력

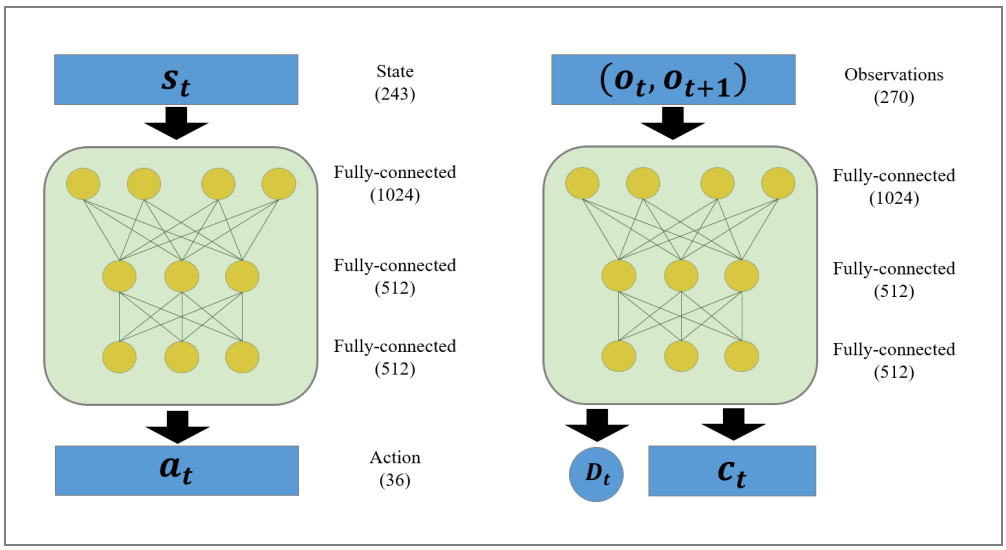

그림 5.2: Policy network(좌)와 Discriminator network(우)의 구성

$\bar{c}_t$를 받은 뒤 액션 $\pi(a_t|s_t, \bar{c}_t) = \mathcal{N}(\mu(s_t, \bar{c}_t), \Sigma)$를 정규 분포 형태로 매핑한다. $\mu(s_t, \bar{c}_t)$는 정규분포의 평균이고 [1024,512,512]개의 유닛으로 구성된 3개의 레이어을 가진 fully-connected 네트워크의 결과로 나오는 값이다. Σ는 정규분포의 표준편차로 모든 값이 0.01로 고정된 대각 공분산 행렬이다. Discriminator $D(s_t, s_{t+1})$는 [1024,512,512]개의 유닛으로 구성된 3개의 레이어을 가진 fully-connected 네트워크로 설계되었다. Discriminator는 Observation을 입력으로 받아 진위를 구분하는 확률 $P(s_t = real|s_t, s_{t+1})$와 캐릭터 모션을 분류하는 확률 분포 $P(c_t|s_t, s_{t+1})$를 출력한다. 모든 네트워크는 활성화 함수로 Rectified Linear Activation Unit(ReLU)함수를 사용했다. 추가적으로, 모든 레이어의 파라미터들에 대해서 Spectral Normalization을 시행했다 [31].

제 6 장 실험 결과

우리는 그림 6.1.처럼 캐릭터가 유저의 명령에 맞게 다양한 시나리오 속의 모션을 수행하는 컨트롤러를 만들었다. 우리는 우리의 캐릭터 컨트롤러가 유저 명령에 맞게 캐릭터가 다양한 액션을 수행할 수 있음을 입증하기 위해 여러가지 실험을 구성했다. 먼저 우리는 학습해야할 모션의 갯수가 늘어나도 control policy π가 학습을 안정적으로 할 수 있음을 보여 캐릭터가 다양한 액션을 수행할 수 있음을 보일 것이다. 그 다음은 학습해야할 모션의 갯수가 늘어나도 discriminator가 캐릭터의 모션을 모션캡쳐 데이터와 잘 구분해내고 모션 유형도 잘 구분해낼 수 있음을 보인다. 이로써 학습된 discriminator를 기반으로 캐릭터가 유저 명령에 따라 액션을 다양하게 수행할 수 있는지를 증명한다. 추가적으로 모션 유형을 전환하는 과정에서 액션과 액션 중간의 포즈가 모션캡쳐 데이터에 없어도 자연스럽게 수행하기 위해 적용한 control reward $r^C(o_t, o_{t+1}, \bar{c}_A, \bar{c}_B,)$가 의미가 있는지도 실험을 통해 입증한다.

6.1 캐릭터의 모션 학습

캐릭터가 모션캡쳐 데이터 속의 모션을 성공적으로 학습했는지 평가하기 위해서는 캐릭터의 모션과 모션캡쳐 데이터의 모션의 차이를 구해야 한다. 하지만 GAIL 기반의 모션 학습은 특정 프레임에 수행해야하는 포즈가 정해져 있지 않기 때문에 동일 모션을 수행했을 때 소요되는 프레임이나 타이밍이 모션캡쳐 데이터와 다를 수 있다. 따라서 우리는 Dynamics Time Warping(DTW)를 사용하여 단일 모션 클립에 대해서 데이터와 캐릭터의 모션의 포즈 차이를 계산했다 [40]. DTW를 활용해서 캐릭터의 현재 모션과 해당 타이밍에서 모션 클립에서의 포즈를 짝지었을 때 두 모

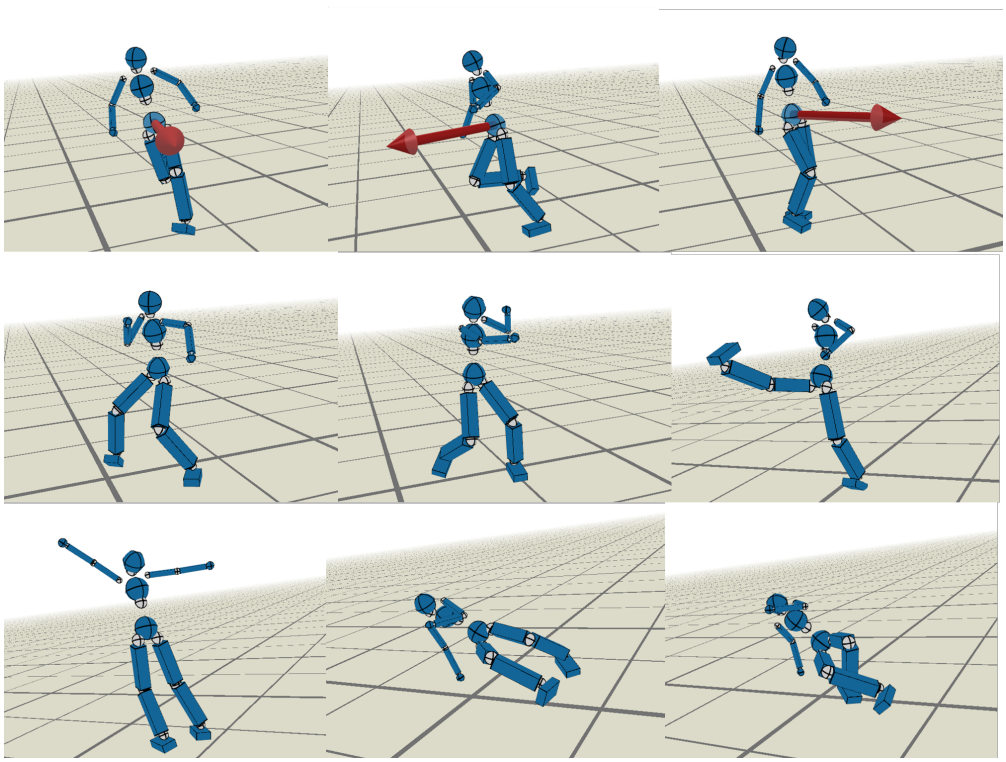

그림 6.1: 유저의 명령에 따라 모션을 수행하는 여러 시나리오들: Walker(위), Fighter(중간), Dancer(아래)

션의 포즈 차이를 계산한 수식은 아래와 같다.

$$e_t^{pose} = \min_{\mathcal{M}} \|(\hat{q}^{-1}q)\|_2, \tag{6.1}$$

q는 3차원 상에서의 모든 조인트의 각도들을 각각 쿼터니언으로 표현한 것이고 $\hat{q}_j$는 모션캡처 데이터에서의 조인트 각도들이다. 즉 e_t^{pose}는 시뮬레이션 프레임 t에서 캐릭터의 각 조인트 각도들을 모션 클립 데이터와 비교한 차이의 l2-norm이다.

6.1.1 모션 클립 학습 결과

우리는 하나의 Control policy를 가지고 10개의 모션들을 학습했다. 10개의 모션들의 대한 정보는 그림 6.2에 있다. 학습 결과를 평가하기 위해 우리는 GAIL 기반의 모션 학습 중에서 Peng et al [26]의 Adversarial Motion Prior과 학습 성능을 비교했다. 동일하게 10가지의 모션을 1.5억개의 샘플을 가지고 학습한 뒤에 모션캡쳐 데이터와의 포즈 차이를 계산했다. 표 6.1은 각 모션별로 각 프레임 별 평균 pose error를 계산한 결과이다.

학습해야할 모션의 수가 한 가지를 넘어 다양한 모션을 학습하지만 모션캡쳐 데이터와의 모션 차이가 대조군인 Adversarial Motion Prior [26]보다 작은 것을 알 수 있다. Punch 동작은 Adversrial Motion Prior가 더 오차가 작지만 실제로 캐릭터의 모션을 확인해보면 punch를 수행하면 포즈 오차가 커져 punch를 학습하지 못하고 준비동작에서 멈춰있다. 따라서 우리의 시스템이 다양한 동작을 학습할 때에도 캐릭터의 수행능력이 떨어지지 않음을 볼 수 있다.

6.1.2 모션의 수에 따른 Policy 학습 결과

캐릭터에게 학습시킬 모션의 수가 늘어날 수록 control policy가 학습해야 하는 모션의 범위가 넓어지므로 학습이 어려워진다. 그림 6.3는 학습할 모션의 수를 각각 1개, 3개, 6개, 10개로 주고 control policy를 학습시켰을 때 학습량에 따른 reward의 양상

표 6.1: 캐릭터가 10개의 모션을 학습한 뒤 수행했을 때 각 모션별 프레임 당 평균 포즈 오차

Motion type	AMP	Ours
Walk	2.462	1.456
Run	2.162	0.808
Jump	2.984	1.781
Breakdance	2.984	2.233
Spindance	1.397	1.051
Uppercut	1.307	0.659
Left punch	1.125	1.412
Right punch	0.881	1.145
Right kick	1.942	1.367
Left kick	2.301	1.672

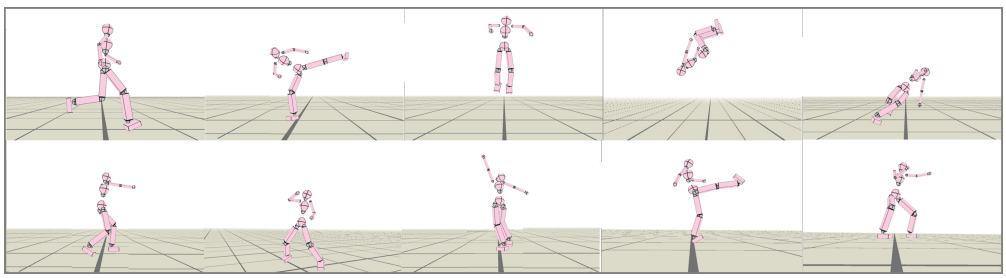

그림 6.2: 학습에 사용되는 레퍼런스 모션들. 왼쪽 위부터 순서대로 Walk, Sidekick, Jump, Backflip, Breakdance, Right punch, Upper cut, Spindance, Right kick, Left jab 이다.

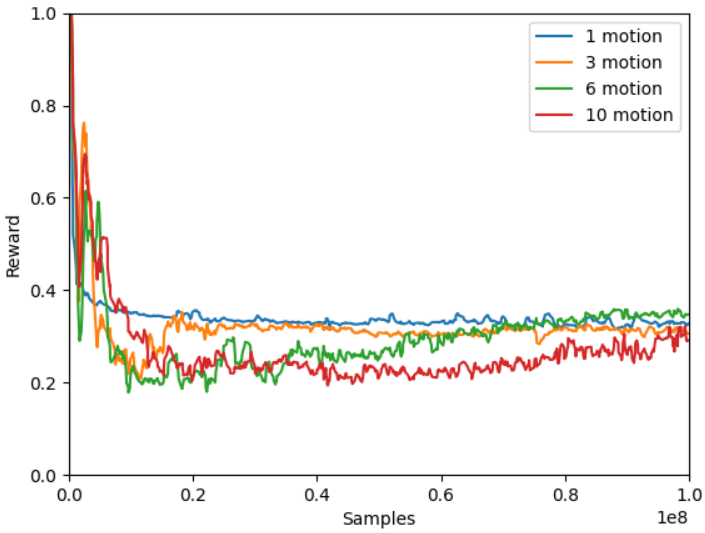

그림 6.3: 학습 모션의 갯수에 따른 policy의 reward 그래프

을 보여준다. 각각 총 1억개의 샘플을 모은 뒤 비교했다. 초반 policy의 reward 양상은 매우 높은 reward를 받다가 점차 낮아진다. 이 양상은 학습 초반에 Discriminator가 캐릭터의 모션을 모션캡쳐 데이터로부터 잘 구분하지 못함을 의미한다. 학습 초반을 넘어 Discriminator가 주어진 모션들을 잘 구분하기 시작하면 reward는 점차 특정한 범위 내에서 수렴하게 된다. reward가 특정 값에 수렴하면 discriminator의 분류 성능과 policy의 사실적인 모션을 만드는 성능이 서로 균형점을 이루면서 대립하면서 학습하고 있음을 의미한다. 그림 6.3을 봤을 때 모션 1개를 학습할 때 가장 빠른 속도로 수렴하게 되고 모션의 갯수가 늘어날 수록 수렴에 필요한 시간이 오래 걸린다. 우리의 control policy는 모션의 갯수를 10개로 늘렸을 때도 reward를 봤을 때 수렴에 이르는 학습량이 많아질뿐 모션의 갯수가 적을 때와 비슷한 수치로 reward로 점차 수렴해나감을 알 수 있다. 결론적으로 우리의 control policy는 적대적으로 학습함에도 불구하고 모션의 갯수에 상관없이 안정적으로 학습할 수 있다.

6.1.3 Discriminator 학습 결과

Discriminator도 마찬가지로 모션의 수가 늘어날 수록 캐릭터의 모션과 모션캡쳐 데이터를 구분하는 학습이 어려워진다. 우리는 기존의 GAIL의 학습방식에서 discriminator에 condition을 추가했다. 따라서 conditional을 부여한 학습방식이 이전 방식과 어떻게 다른 결과가 나타나는지 비교했다. 그림 6.4는 discriminator에 condition을 부여하지 않는 기존의 GAIL의 discriminator [26]와 우리의 conditional discriminator의 성능을 2억개의 샘플을 수집한 후 비교했다. 그 결과 모션의 수가 많을 때 우리의 discriminator가 모션캡쳐 데이터와 캐릭터의 모션에 대해 모두 분류 성능이 좋은 것을 볼 수 있다. 그림 6.5는 모션 캡쳐 데이터와 캐릭터의 모션을 구분하기 위한 style loss값을 비교한 결과로 우리의 discriminator의 style loss가 더 낮은 값으로 수렴하고 있는 것을 볼 수 있다.

그 다음 classification loss L_{class}도 우리가 사용한 loss term과 대조군인 least square loss term과 비교했다. least square loss term은 아래와 같다.

$$L_{class,LS} = \sum_{t=0}^{T} \|\bar{c}_t - C(o_t, o_{t+1})\|_2 \tag{6.2}$$

그림 6.6과 그림 6.7는 본 논문에서 classification loss로 사용한 cross entropy loss와 수식 6.2의 least square loss를 비교한 결과이다. 그림 6.6을 보면 cross entropy loss가 가파르게 낮아지며 더 낮은 loss 값으로 수렴함을 알 수 있고 그림 6.7를 보면 discriminator가 모션캡쳐 데이터와 캐릭터 모션을 더 잘 구분해내 높은 정확도를 보인다.

6.2 모션 분류기를 이용한 캐릭터 컨트롤

캐릭터의 모션을 생성하는 policy π는 사실적인 모션을 만들수록 높은 Reward를 받지만 유저의 의도대로 행동했을 때도 높은 Reward를 받는다. Policy π가 생성한

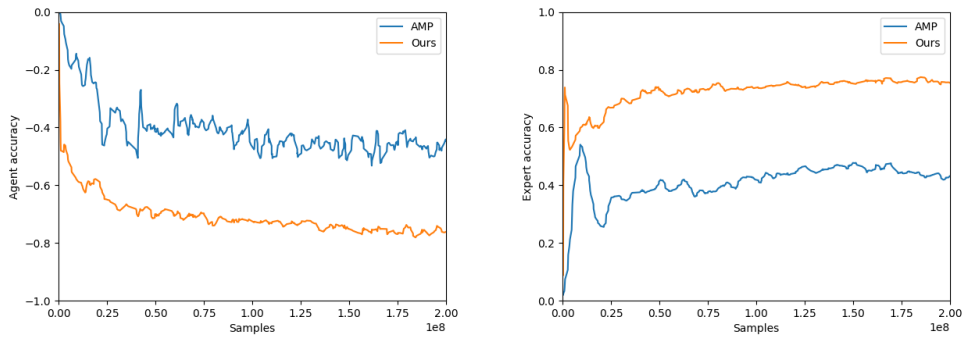

그림 6.4: Discriminator에 condition을 부여함에 따른 discriminator의 분류성능 전후 비교

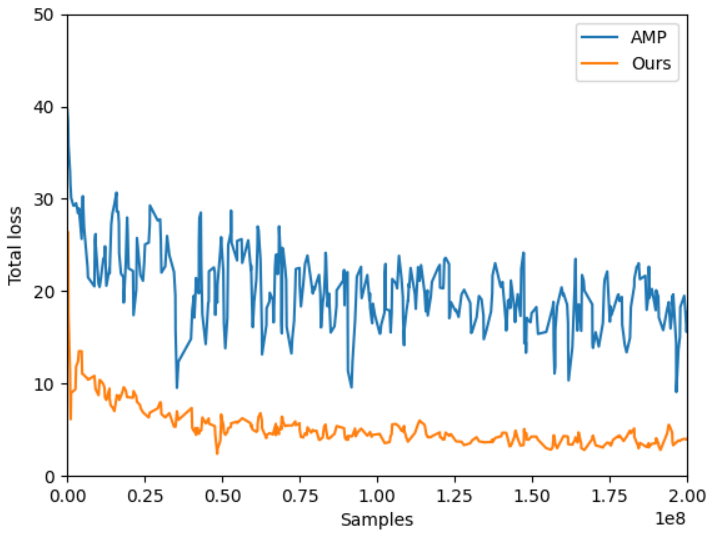

그림 6.5: Discriminator에 condition을 부여함에 따른 discriminator의 loss 전후 비교

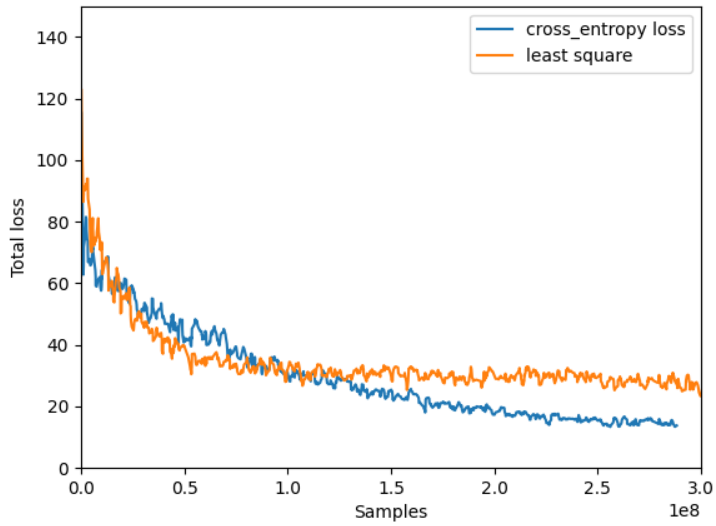

그림 6.6: Classification Loss 형태에 따른 discriminator의 loss 양상

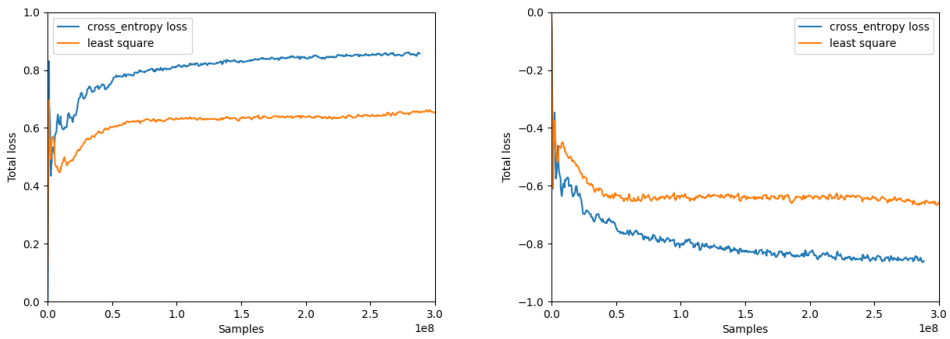

그림 6.7: Classification Loss 형태에 따른 모션캡쳐 데이터와 캐릭터 모션에 대한 discriminator의 분류 정확도

모션이 유저가 명령한 모션인지 평가하기 위해서는 먼저 Discriminator의 모션 분류기가 성공적으로 모션을 분류한다는 전제가 있어야 한다. 따라서 이번 섹션은 모션 분류기의 성능을 입증하고 모션 분류기를 통해 캐릭터가 유저 명령에 맞게 모션을 바꿀 수 있음을 보인다.

6.2.1 모션 분류기의 성능

먼저 모션 분류기의 성능을 검증하려면 다양한 종류의 모션을 올바르게 분류할 수 있는지 증명해야한다. 그림 6.8은 캐릭터가 10개의 모션을 각각 수행했을 때 학습된 discriminator가 각 모션 별로 모션의 유형을 추측한 결과이다. 해당 결과를 얻기 위

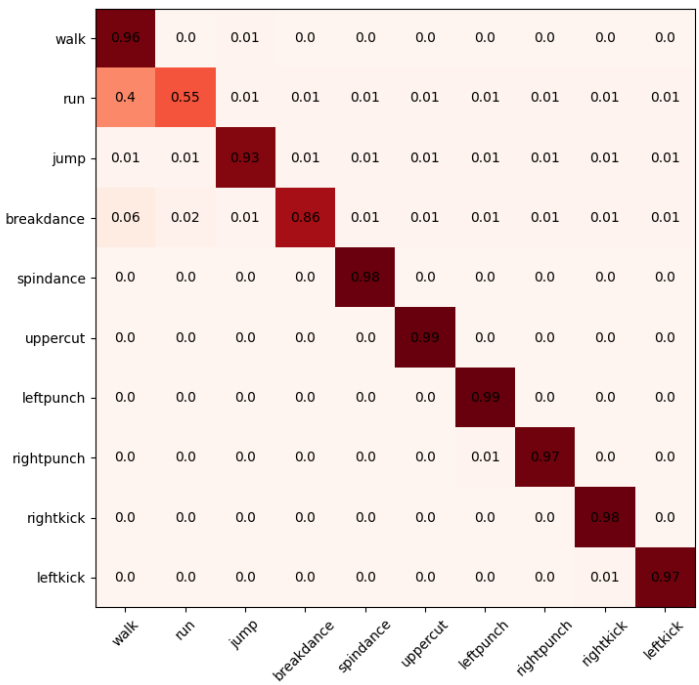

그림 6.8: 10개의 모션 별 모션 분류기의 분류 정확도

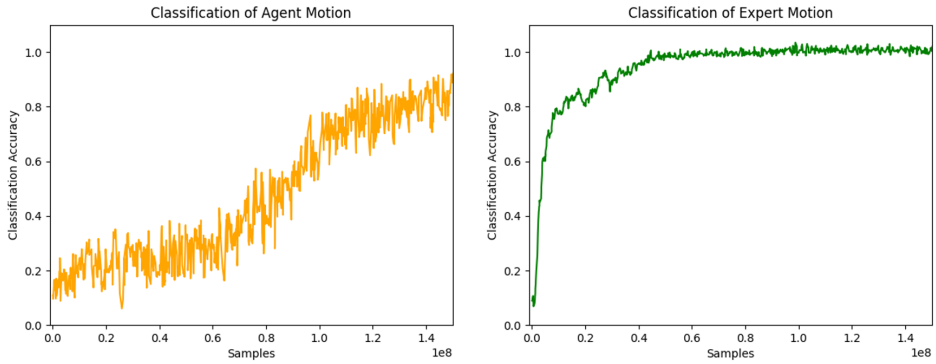

그림 6.9: 캐릭터의 모션(좌)과 모션캡쳐 데이터(우)에 대한 Discriminator에 모션 분류 정확도

해 Discriminator는 총 3 억개의 샘플을 수집했다. 표 안의 수치는 모션 분류기가 y 행의 모션을 x열의 모션으로 추측한 확률로 모두 [0,1]사이의 값을 갖고 각 행마다 모든 열의 값을 합하면 1이된다. 결과를 보면 run과 breakdance를 제외한 모든 모션들에 대해 0.9 이상의 확률로 올바르게 분류하는 것을 알 수 있다. 모션분류기가 run 모션을 walk 모션으로 혼동할 확률이 0.4정도 있는데, 이는 데이터 수집과정에서 run 데이터가 뛰는 것뿐만아니라 걷는 속도로 움직이는 모션들도 같이 수집되면서 생긴 현상으로 보인다. breakdance의 경우 walk로 혼동할 확률이 0.06정도 되지만 분류 성능에 문제가 생길정도의 큰 수치는 아니고 control policy가 모션을 학습하는 과정에서 생긴 현상으로 볼 수 있다.

학습량에 따라 모션 분류 정확도가 얼마나 개선되는지는 그림 6.9를 통해 알 수 있다. 그림 6.9는 모션 분류기가 캐릭터의 모션(왼쪽 그래프)과 모션캡쳐 데이터(오른쪽 그래프)의 모션에 대해서 각각 모션 유형을 얼마나 잘 분류하는지 학습량에 따라 나타낸 것이다. 학습은 샘플 1.5억개가 모일때까지 진행하고 결과를 보았다. 모션캡쳐 데이터에 대해서 모션 분류기는 비교적 빠르게 분류 정확도가 증가하는 것을 볼 수 있고 100%의 정확도로 수렴해가는 것을 볼 수 있다. 반면 캐릭터의 모

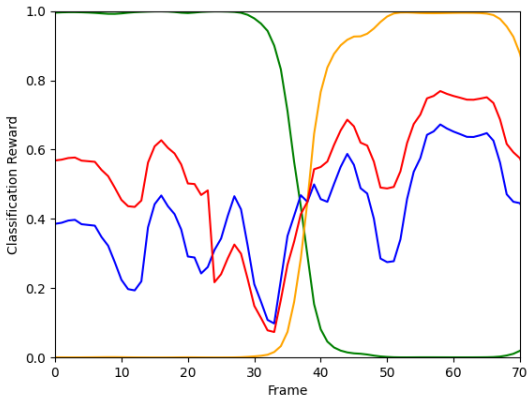

그림 6.10: 캐릭터가 right kick 도중 left kick 명령을 받았을 때 Discriminator의 모션 분류 결과 및 style reward

션에 대해서는 비교적 천천히 증가해나가는 양상을 볼 수 있다. 이를 토대로 봤을 때 모션 분류기는 모션캡쳐 데이터를 통해 먼저 모션 유형들을 학습하며 구분한다고 볼 수 있다. 모션캡쳐 데이터를 잘 구분하기 전까지는 캐릭터의 모션을 분류하기 어렵고 모션캡쳐 데이터를 잘 구분하기 시작했을 때 캐릭터의 모션들을 점차 정확하게 분류하기 시작한다. 학습 초기에는 캐릭터의 모션이 부정확하기 때문에 모션 분류기가 제대로 분류를 하지 못하다가 control policy가 점차 사실적인 모션을 생성하면서 모션 분류기의 정확도가 높아진다. 결과적으로 학습량을 더 늘리면 그림 6.8처럼 모션 분류기가 높은 정확도를 가지고 모션들을 분류할 수 있다.

6.2.2 캐릭터 제어 성능

모션 분류기를 통해 Discriminator의 성능을 향상시키는 것도 있지만 control reward r^C를 통해 캐릭터가 유저 명령에 맞는 동작을 수행하는 것이 모션 분류기의 주 목적이다. 그림 6.10는 캐릭터가 right kick 이후 left kick을 수행했을 때 discriminator가 캐릭터의 모션을 right kick으로 분류하는지와 left kick으로 분류하는지에 대해 확률

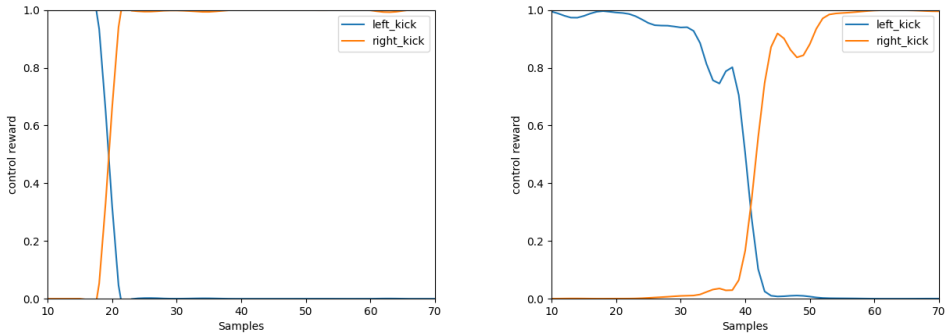

그림 6.11: Transition phase가 없을 때(좌)와 있을 때(우) 모션 전환 상황에서의 각 모션에 대한 control reward

로 표현한 것이다. 유저가 명령을 바꾸는 순간 right kick으로 분류할 확률이 서서히 낮아지면서 left kick으로 분류할 확률이 점차 증가하는 것을 볼 수 있다. 추가적으로 모션을 전환하는 과정에서 style reward r^S와 total reward r이 어떻게 나타나는지도 볼 수 있는데, 전반적으로 값이 일정치 않아 일반화할 수는 없지만 유저 명령이 바뀌었을 때 reward가 낮아졌다가 전환이 끝나면서 점차 reward가 높아지는 것을 볼 수 있다. 모션 전환 과정에서 모션캡쳐 데이터에 없는 동작을 수행하지만 점차 유저의 명령에 맞는 자연스러운 모션을 실시한다.

6.3 모션 전환 성능

본 논문은 유저가 명령하는 액션이 변했을 때 캐릭터에게 transition phase w_{trans}를 주어 천천히 모션을 전환하도록 했다. w_{trans}는 60프레임 동안 0에서 1로 일정하게 변함으로써 두 액션 사이의 중간 동작을 자연스럽게 유도한다. 이를 입증하기 위해 w_{trans}가 없는 경우와 w_{trans}가 없지만 순차적으로 이전 액션과 현재 액션을 블렌딩하는 경우, 그리고 우리의 알고리즘을 비교했다.

그림 6.11은 유저의 명령이 바뀌어 캐릭터가 Right kick에서 Left kick으로 동작을 전환할 때 control reward 양상을 나타낸다. Transition phase없이 동작을 전환할 때는

약 5프레임 안에 급격하게 동작을 전환하는 것을 볼 수 있다. 이런 경우 모션이 부자연스러운 결과가 나타난다. Transition phase를 사용해 두 액션 사이를 linear하게 전환한 경우 전환하는 과정이 상대적으로 길어지는 것을 볼 수 있다. 따라서 갑작스럽지 않게천천히 동작을 전환할 수 있다.

제 7 장 고찰 및 결론

우리는 물리 시뮬레이션 상에서 유저의 명령에 따라 실시간으로 다양한 행동을 수행할 수 있는 캐릭터 컨트롤러를 고안했다. 우리 접근 방식의 핵심은 유저 컨트롤러와 캐릭터의 모션 학습을 별도로 두지 않고 함께 학습할 수 있다는 점이다. 따라서 두 가지 시스템을 별도로 만드는 수고로움과 유저 컨트롤러를 만들 때 필요한 전처리 과정을 요구하지 않는다.

또한 유저의 명령에 맞게 캐릭터가 움직이기 위해 control policy를 학습하는 과정에서 reward 설계가 복잡하지 않다. 우리의 reward 설계는 최종 reward를 이루는 두 개의 reward 모두 discriminator에서 나온 결과값을 가지고 만들어진다. 따라서 기존의 모방학습 기반의 복잡한 reward 설계보다 단순하고 모션에 따라 reward 구성이나 변수 설정이 달라지지 않는다.

우리의 알고리즘은 캐릭터가 배워야할 모션들 중 역동적인 모션들이 많을 수록 이점이 있다. 역동적인 모션들의 모션캡쳐 데이터들은 보통 단일 모션 클립들로는 존재하지만 역동적인 모션들이 서로 연계되어 이어지는 모션들은 많지 않다. 우리 시스템은 다양한 모션 유형들을 한번에 수행할 때 모션 유형 간 전환 동작들이 모션캡쳐 데이터에 없더라도 캐릭터가 스스로 생성할 수 있다.

우리의 물리 기반 캐릭터 컨트롤러는 단일 동작을 너머 다양한 동작들을 캐릭터에게 명령할 수 있다. 향후 연구는 다양한 동작의 범위를 더 넓혀 10가지 동작을 넘어 30, 40, N가지 동작을 실시간으로 캐릭터에게 명령을 내릴 수 있는 시스템을 구축하는 것이다. 이를 위해 가장 먼저 해결해야할 점은 학습 효율성이다. 현재 10가지 동작을 학습하기 위해서는 동작의 유사도와 관계없이 1가지 동작을 배울 때보다 10배의 시간이 필요하다. 동작의 가짓수가 늘어나면 이 또한 기하급수적으로 늘어날텐데 이를 효율적으로 해결해 빠르게 학습할 수 있는 알고리즘이 필요하다. 또한 모션이 좀더 모션캡쳐데이터에 가까운 사실적인 모션을 수행할 수 있도록 개선된

필요가 있다. 동작의 가짓수가 늘어날수록 캐릭터가 학습해야할 동작의 범위가 늘어나기 때문에 모션의 정확도가 전반적으로 떨어진다. 이를 해결하기 위한 별도의 알고리즘도 추후 연구 주제가 될 수 있을 것이라고 기대한다. 따라서 control policy를 학습을 개선하기 위해 학습의 효율을 개선할 수 있는 pre-trained teacher policy에 대한 개념을 구상하고 있다. 만약 모션의 가짓수가 100개가 되어도 필요한 학습량이 효율적으로 줄어든다면 우리의 캐릭터 컨트롤러가 좀더 확장성있는 시스템이 될 것이다.

Curriculum learning도 효율적인 학습의 예가 될 수 있다. 꼭 다양한 모션을 학습하는 과정과 유저의 명령에 맞게 모션을 취하는 과정을 한번에 학습할 필요는 없다. 먼저 단일 모션들을 수행하는 방법들을 각각 학습한 뒤 유저에 명령에 맞게 모션을 전환하는 과정을 이후에 학습하면서 학습가능 한 모션의 범위를 확장해나갈 수 있다.

참고 문헌

[1] Jessica K Hodgins, Wayne L Wooten, David C Brogan, and James F O'Brien. Animating human athletics. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 71–78, 1995.

[2] Jessica K Hodgins and Nancy S Pollard. Adapting simulated behaviors for new characters. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 153–162, 1997.

[3] KangKang Yin, Kevin Loken, and Michiel Van de Panne. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics*, 26(3), 2007.

[4] Yao-Yang Tsai, Wen-Chieh Lin, Kuangyou B Cheng, Jehee Lee, and Tong-Yee Lee. Real-time physics-based 3d biped character animation using an inverted pendulum model. *IEEE transactions on visualization and computer graphics*, 16(2):325–337, 2009.

[5] Jack M Wang, David J Fleet, and Aaron Hertzmann. Optimizing walking controllers. pages 1–8, 2009.

[6] C Karen Liu and Zoran Popović. Synthesis of complex dynamic character motion from simple animations. *ACM Transactions on Graphics (TOG)*, 21(3):408–416, 2002.

[7] Anthony C Fang and Nancy S Pollard. Efficient synthesis of physically valid human motion. *ACM Transactions on Graphics (TOG)*, 22(3):417–426, 2003.

[8] Alla Safonova, Jessica K Hodgins, and Nancy S Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics (ToG)*, 23(3):514–521, 2004.

[9] Kevin Wampler and Zoran Popović. Optimal gait and form for animal locomotion. *ACM Transactions on Graphics (TOG)*, 28(3):1–8, 2009.

[10] Victor Brian Zordan and Jessica K Hodgins. Motion capture-driven simulations that hit and react. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 89–96, 2002.

[11] Shinichiro Nakaoka, Atsushi Nakazawa, Kazuhito Yokoi, Hirohisa Hirukawa, and Katsushi Ikeuchi. Generating whole body motions for a biped humanoid robot from captured human dances. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 3, pages 3905–3910. IEEE, 2003.

[12] Kwang Won Sok, Manmyung Kim, and Jehee Lee. Simulating biped behaviors from human motion data. *ACM Transactions on Graphics*, 26(3), 2007.

[13] Marco Da Silva, Yeuhi Abe, and Jovan Popović. Simulation of human motion data using short-horizon model-predictive control. In *Computer Graphics Forum*, volume 27, pages 371–380, 2008.

[14] Uldarico Muico, Yongjoon Lee, Jovan Popović, and Zoran Popović. Contact-aware nonlinear control of dynamic characters. In *ACM SIGGRAPH 2009 papers*, pages 1–9. 2009.

[15] Yoonsang Lee, Sungeun Kim, and Jehee Lee. Data-driven biped control. *ACM Transactions on Graphics*, 29(4), 2010.

[16] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics*, 37(4), 2018.

[17] Seunghwan Lee, Moonseok Park, Kyoungmin Lee, and Jehee Lee. Scalable muscle-actuated human simulation and control. *ACM Transactions on Graphics*, 38(4), 2019.

[18] Kevin Bergamin, Simon Claver, Daniel Holden, and James Richard Forbes. Drecon: Data-driven responsive control of physics-based characters. *ACM Transactions on Graphics*, 38(6), 2019.

[19] Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. Physics-based motion capture imitation with deep reinforcement learning. In *Proceedings of the 11th annual international conference on motion, interaction, and games*, pages 1–10, 2018.

[20] Soohwan Park, Hoseok Ryu, Seyoung Lee, Sunmin Lee, and J. Lee. Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics*, 38(6), 2019.

[21] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics*, 39(4), 2020.

[22] Xue Bin Peng, Glen Berseth, Kangkang Yin, and Michiel Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics*, 36(4), 2017.

[23] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[24] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[25] Josh Merel, Yuval Tassa, Dhruva TB, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017.

[26] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)*, 40(4):1–20, 2021.

[27] Pei Xu and Ioannis Karamouzas. A gan-like approach for physics-based imitation learning and interactive character control. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 4(3):1–22, 2021.

[28] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[29] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651. PMLR, 2017.

[30] Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[31] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[32] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

[33] Stefan Schaal, Auke Ijspeert, and Aude Billard. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 2003.

[34] Alla Safonova, Nancy Pollard, and Jessica K Hodgins. Optimizing human motion for the control of a humanoid robot. *Proc. Applied Mathematics and Applications of Mathematics*, 2003.

[35] Jie Tan, Karen Liu, and Greg Turk. Stable proportional-derivative controllers. *IEEE Computer Graphics and Applications*, 2011.

[36] Faraz Torabi, Garrett Warnell, and Peter Stone. Adversarial imitation learning from state-only demonstrations. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2229–2231, 2019.

[37] Liyiming Ke, Sanjiban Choudhury, Matt Barnes, Wen Sun, Gilwoo Lee, and Siddhartha Srinivasa. Imitation learning as f-divergence minimization. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 313–329. Springer, 2020.

[38] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2794–2802, 2017.

[39] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

[40] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

ABSTRACT

Physics-based Character Controller with Diverse Motor Skills

Yongwoo Lee

Department of Computer Science and Engineering

The Graduate School

Seoul National University

The goal of this research is to control a physics-based character which learns several dynamic motor skills. When a user commands a specific motion to the character, the character can design a motion plan to perform the given motion from current pose. We present a network-based learning algorithm that learns various motor skills and transitioning motions between the motor skills from disparate motion clip datasets. The overall framework for our controller is composed of a DeepRL-based controller which generates a character's behavior, and a discriminator which induces the controller to produce proper motions from a user's commands. The discriminator and the controller take outputs from each other as input and improve each performance through adversarial training process. This paper demonstrates the effectiveness of this approach through examples with an interactive character that learns various dynamics motor skills and follow a user command in physics simulation.

Keywords: Computer Graphics, Character Animation, Physics Simulation, Deep Reinforcement Learning, Adversarial Learning

Student Number: 2020-22001

감사의 글

코로나19 바이러스와 함께 시작했던 대학원 생활은 위드코로나 시대와 함께 마무리 짓게 되었습니다. 석사 과정 2년은 정신없이 지내다보니 정말 짧은 시간이었고 그럼에도 불구하고 많은 일이 있었습니다. 2년 동안 어려운 일들이 많았지만 혼자라면 어려운 일도 함께 했기에 극복할 수 있었습니다. 대학원 생활에 적응할 수 있도록 도와주고 무엇이든 알려주었던 연구실 일원들에게 감사드립니다. 무엇보다 힘든 순간들이 있을 때마다 함께 있어주어 감사했습니다.

석사 과정에 진학하기까지 진로에 있어 많은 내적 갈등이 있었고 진학 이후에도 많은 고민이 있었습니다. 그럴 때마다 제가 불안해하지 않고 안정감을 느낄 수 있었던 건 뒤돌아 섰을 때 항상 그 자리에 계셨던 가족 덕분입니다. 항상 제 이야기를 들어주시고 든든한 버팀목이 되어주신 아버지,어머니, 형님들, 형수님께 감사드립니다.