**< 캡스톤디자인 최종보고서 >**

프로젝트 명: 사용자와 교감하는 대화형 인공지능을 위한 백채널 예측 모델 개발

팀원(참여학생): 이재형(소프트웨어융합학과), 최진욱(소프트웨어융합학과)

**요 약**

# 서론

## 연구배경

## 연구 목표

## 기대 효과

# 배경 지식 및 관련 연구

## 합성곱 신경망(Convolutional Neural Networks, CNN)

## 완전 연결 계층(Fully Connected Layer, FC Layer)

## 손실함수(Loss Function)

## 기존 연구 문제점 및 해결 방안

# 추진 내용

## 팀 구성 및 역할

|  |  |  |
| --- | --- | --- |
| 이름 | 역할 | 수행업무 |
| 이재형 | 팀장 | Motion matching 알고리즘 구현, 게임 로직 구현, 일정 및 리소스 관리 |
| 최진욱 | 팀원 | Motion matching 알고리즘 구현, 게임 로직 구현, 일정 및 리소스 관리 |

## 전체 시스템 구성

이 프로젝트는 모션 캡처 데이터인 BVH 파일에서 추출한 애니메이션 데이터를 기반으로 한 데이터셋 구축과, PyTorch를 이용하여 구현한 Projector, Stepper, Decompressor 네트워크를 통해 언리얼 엔진에서의 모션매칭 시스템을 개발하는 것을 주요 목표로 합니다. 데이터셋 구축 단계에서는 BVH 파일의 관절 위치와 회전을 처리하고, 좌우 반전과 같은 데이터 증강 기법을 적용하여 다양한 pose 데이터를 생성합니다. 이 데이터는 60fps로 변환되고, 보간을 통해 부드럽게 처리됩니다.

Decompressor는 특정 프레임의 feature vector를 입력으로 받아 자세를 생성하며, Stepper는 현재 프레임의 feature vector와 latent 변수를 이용하여 다음 프레임을 예측합니다. Projector는 주어진 쿼리 벡터로부터 가장 적합한 매칭 피처 벡터를 생성하여, 캐릭터의 자연스러운 걷기 혹은 달리기 모션을 얻습니다. 메인 시스템은 언리얼 엔진의 좌표계와 BVH 데이터의 호환성을 유지하기 위해 좌표계를 통합하고, 사용자의 입력 처리를 담당하며 이를 기반으로 모션 매칭을 실행하여 자연스러운 애니메이션을 출력합니다.

## 핵심 기능 개발

(데이터셋 구축 프로그램 구현)

BVH (Biovision Hierarchy) 파일 형식으로 된 모션 캡처 데이터를 처리하여 애니메이션 데이터베이스를 생성하는 작업을 수행합니다. BVH 파일은 보통 인체의 관절 위치와 회전을 프레임 단위로 저장하는데, 이를 불러와 다양한 처리를 한 후 바이너리 파일로 저장합니다.

Data augmentation을 위해서 주어진 애니메이션 데이터를 좌우 반전합니다. 각 관절의 위치와 회전을 미러링하여 반환하고 이를 기존 데이터에 추가합니다.

처리할 BVH 파일 목록을 다음과 같이 정의하고, 데이터베이스에 필요한 리스트들을 초기화합니다.

bone\_positions, bone\_velocities, bone\_rotations, bone\_angular\_velocities, bone\_parents, bone\_names, range\_starts, range\_stops, contact\_states

이후 데이터를 60fps로 변환하고, cubic 보간을 통해 데이터를 부드럽게 만듭니다. 또한 Simulation Bone 정보를 추출하여 저장합니다. 최종적으로 처리된 모든 데이터를 frame 단위로 구분되는 하나의 배열로 결합한 후 바이너리 파일로 저장합니다.

(Projector, stepper, decompressor 학습 코드 내용)

Projector, stepper, decompressor는 pytorch를 이용해 구현했습니다.

Decompressor 설명

**Decompressor**의 주된 목표는 특정 프레임의 feature vector를 받아서 대응하는 자세를 직접 생성하는 것입니다. 피처 벡터는 일반적으로 대응하는 자세에 대한 중요한 정보를 포함하지만, 완전히 재구성하기에는 정보가 부족할 수 있습니다. 이를 보완하기 위해 추가적인 잠재 변수를 도입합니다. 이러한 구조는 **Autoencoder**와 유사합니다.

**Compressor** 네트워크를 사용하여 pose 정보 를 저차원 표현 로 변환합니다. 이 를 와 연결한 후, Decompressor에 입력으로 제공하여 원래의 자세 ​를 복원합니다. 이를 통해 피처 벡터 에서 누락된 추가 정보를 에 인코딩하는 방법을 학습합니다.

단순한 평균 제곱 오차(Mean Squared Error, MSE) 손실을 사용하면 떨림이 있는 모션이 발생할 수 있습니다. 대신, 시각적으로 인식되는 오류를 최소화하기 위해 다음과 같은 **Forward Kinematics**를 사용하여 손실함수를 설계합니다.

Stepper 설명

Stepper 네트워크는 현재 프레임의 피처 벡터와 잠재 변수에서 다음 프레임의 피처 벡터와 잠재 변수를 예측하는 역할을 합니다. 이 네트워크는 주어진 프레임의 피처 벡터 와 잠재 변수 ​를 입력으로 받아서, 다음 프레임의 피처 벡터 ​와 잠재 변수 ​를 생성합니다. 이를 위해 현재 프레임의 피처 벡터와 잠재 변수에 델타 값을 더하는 방식으로 다음 프레임의 벡터를 예측합니다.

Stepper 네트워크는 다음과 같은 방식으로 훈련됩니다:

**Auto-regressive 훈련**: 짧은 길이의 피처 벡터 시퀀스 와 잠재 변수 를 사용하여, 네트워크가 다음 피처 벡터와 잠재 변수를 반복적으로 예측하고 이를 다음 프레임의 입력으로 사용합니다.

**손실 함수**: 손실 함수는 여러 요소를 고려하여 구성되며, 각 손실 항목에 대해 균등한 가중치를 줍니다.

**미니 배치 훈련**: 미니 배치 내의 모든 샘플에 대해 적용하여 평균 결과를 통해 네트워크 매개변수 를 업데이트합니다.

Projector 설명

Projector는 가장 가까운 이웃 검색 기능을 에뮬레이트하여, query vector 와 가장 가까운 항목 을 에서 찾는 대신, 직접 매칭되는 피처 벡터와 잠재 변수를 생성합니다.

매칭 데이터베이스의 피처 벡터 를 주어진 노이즈 크기 ​로 스케일된 가우시안 노이즈 벡터 을 추가하여 쿼리 벡터 를 생성합니다. 이 쿼리 벡터 로부터 최근접 이웃 를 찾습니다. Projector 네트워크는 와 ​를 출력하도록 훈련됩니다. 미니 배치 내의 모든 샘플에 대해 적용하고 평균 결과를 통해 네트워크 매개변수 를 업데이트합니다. 다양한 노이즈 크기를 샘플링하여, Projector가 다양한 크기의 변형에 대해 견고하도록 만듭니다. 손실 항목에 대해 균등한 가중치를 부여합니다.

Projector, stepper, decompressor를 학습시키고 이를 통해 매칭 데이터베이스에 의존하지 않고도 기능을 수행할 수 있습니다. 구체적인 절차는 다음과 같습니다:

1. **사용자 쿼리 처리**:
   * 매 프레임마다 사용자 쿼리 를 Projector에 전달하여, 가장 가까운 피처 벡터 와 잠재 변수 ​​를 찾습니다.
2. **Stepper와 Decompressor 사용**:
   * 각 프레임마다 Stepper를 사용하여 매칭 피처 벡터와 잠재 변수를 갱신합니다.
   * 갱신된 피처 벡터와 잠재 변수를 사용하여 Decompressor를 통해 포즈를 생성합니다.

(언리얼 엔진에서 구동되는 모션매칭 시스템 구현)

학습시킨 Projector, stepper, decompressor와 사용자 입력을 기반으로 게임 캐릭터의 애니메이션을 출력하는 모션매칭 시스템을 언리얼 엔진에서 구현했습니다. 핵심 기능들을 간략하게 설명합니다.

* **desired\_gait\_update()**: 이 함수는 현재 캐릭터가 걷기에서 뛰기로 전환할 필요성이 있는지를 결정합니다. 플레이어가 게임패드를 누르면 걷기에서 뛰기로 전환할 수 있습니다.
* **desired\_velocity\_update()**: 게임패드의 입력 및 카메라 방향 등을 고려하여 캐릭터의 원하는 속도와 방향을 계산합니다. 이는 캐릭터가 어디로 이동하고 있는지를 나타내며, 예를 들어 게임패드의 스틱 입력에 따라 전방, 후방, 좌우로의 이동을 결정합니다.
* **desired\_rotation\_update()**: 캐릭터의 원하는 회전과 방향을 계산합니다. 예를 들어, 플레이어가 캐릭터를 특정 방향으로 회전시키는 입력을 하면, 이에 맞춰 캐릭터가 회전하도록 설정됩니다.
* **trajectory\_desired\_rotations\_predict()**: 이 함수는 캐릭터의 예상 회전을 계산합니다. 현재 입력 및 이전 회전 상태를 고려하여 다음에 캐릭터가 어떻게 회전할지 예측합니다.
* **trajectory\_rotations\_predict()**: 이 함수는 캐릭터의 회전을 예측하고, 이전 회전 상태와 원하는 회전 상태를 보간하여 자연스러운 애니메이션 전환을 유도합니다.
* **database\_search()**: 데이터베이스에서 가장 적합한 애니메이션 프레임을 검색합니다. 현재 상태에 가장 적합한 걷기 애니메이션 또는 뛰기 애니메이션 프레임을 찾습니다.
* **projector\_evaluate()**: 이 함수는 현재 애니메이션 상태와 새로운 프로젝션 사이의 평가를 수행하여, 새로운 애니메이션 프레임을 결정합니다. 새로운 애니메이션 프레임이 이전 프레임과 충분히 다르면, 캐릭터의 움직임을 업데이트합니다.
* **Transition:** 현재 프레임과 예상 프레임(프로젝션) 사이의 차이를 평가하여, 적절한 애니메이션 전환을 결정하는 조건입니다.
* **decompressor\_evaluate**: 예상 프레임에 대한 자세를 평가하여 필요한 모든 캐릭터의 뼈 위치 및 회전을 계산합니다.
* **Features\_curr** 및 **Latent\_curr**: 현재 캐릭터의 특징 및 숨겨진 변수를 업데이트하여 다음 프레임의 평가에 사용됩니다.
* **inertialize\_pose\_transition()**: 이 함수는 애니메이션 프레임 간의 자연스러운 전환을 담당합니다. 이전 애니메이션 상태에서 새로운 애니메이션 상태로의 움직임을 부드럽게 전환하여 캐릭터의 움직임이 자연스럽게 보이도록 합니다.
* **stepper\_evaluate()**: 이 함수는 애니메이션 스텝(보행 주기)을 평가하고, 다음 스텝으로 전환합니다.

(이를 이용한 배달 게임 구현)

모션 매칭을 이용해서 게임패드로 입력을 받고, 매우 자연스러운 걷기와 달리기 모션을 출력합니다. 이 시스템을 이용하여 캐릭터가 주어진 목적지로 물건을 배달하는 간단한 게임을 제작하였습니다. 이를 통해 모션 매칭 시스템이 원활하게 작동하는 것을 시각적으로 확인할 수 있습니다.

## 이슈 및 대응

(좌표계 변환 관련)

애니메이션 data인 BVH에서 애니메이션 정보를 추출할 때 공간 좌표계를 오른손 좌표계(Yup, Zforward)로 세팅하였습니다. 그러나 Unreal engine은 공간 좌표계가 왼손 좌표계(Zup, Xforward)로 설정이 되어있습니다. 때문에 이 둘 간의 호환을 위해 좌표계를 변환하는 작업을 진행했습니다.

(캐릭터 밀림 현상과 관련 (simulation object와 character mesh의 연계))

기본적으로 시뮬레이션 객체는 코드에 의해 이동하고, 캐릭터 엔티티(캐릭터 Mesh)는 애니메이션 데이터에 따라 움직입니다. 이 두 이동 방식 간의 차이를 최소화하기 위해 다양한 방법을 활용해야 합니다. 다음 링크의 글을 참고하여 문제를 해결하였습니다.

(Obstacle과 관련)

분량 보고 내용 넣을지 확인하기

# 결과

(유튜브 영상 링크 첨부)

현재 보여지는 결과물은 ‘걷기’와 ‘달리기’ 모션을 학습하여 얻은 것입니다. 학습은 15시간 정도 소요되었습니다.

Loss는 다음과 같습니다.

decompressor -> loss: 1.5141

stepper -> loss: 1.020

projector -> loss: 0.730

사용자의 입력에 적절히 반응하여 매우 자연스러운 애니메이션을 출력하는 것을 확인할 수 있습니다.

# 결론

논문에서의 결론 넣기

앞으로 발전시켜야 할 내용 넣기

참고문헌

[1] D. Ortega, C.-Y. Li and N. T. Vu, "OH, JEEZ! or UH-HUH? A Listener-Aware Backchannel Predictor on ASR Transcriptions,” ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Barcelona, Spain, pp. 8064-8068. 2020.

[2]