특집논문 (Special Paper)
방송공학회논문지 제26권 제5호, 2021년 9월 (JBE Vol.26, No.5, September 2021)
https://doi.org/10.5909/JBE.2021.26.5.519
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

RGB 이미지를 이용한 관절 추정 네트워크와 결합된 FBX 형식 애니메이션 생성 시스템

이 유 진[a], 김 상 준[b], 박 구 만[c]‡

FBX Format Animation Generation System Combined with Joint Estimation Network using RGB Images

Yujin Lee[a], Sangjoon Kim[b], and Gooman Park[c]‡

요 약

최근 게임, 영화, 애니메이션 다양한 분야에서 모션 캡처를 이용하여 신체 모델을 구축하고 캐릭터를 생성하여 3차원 공간에 표출하는 콘텐츠가 증가하고 있다. 마커를 부착하여 관절의 위치를 측정하는 방법에서 촬영 장비에 대한 비용과 같은 문제를 보완하기 위해 RGB-D 카메라를 이용하여 애니메이션을 생성하는 연구가 진행되고 있지만, 관절 추정 정확도나 장비 비용의 문제가 여전히 존재한다. 이에 본 논문에서는 애니메이션 생성에 필요한 장비 비용을 줄이고 관절 추정 정확도를 높이기 위해 RGB 이미지를 관절 추정 네트워크에 입력하고, 그 결과를 3차원 데이터로 변환하여 FBX 형식 애니메이션으로 생성하는 시스템을 제안한다. 먼저 RGB 이미지에 대한 2차원 관절을 추정하고, 이 값을 이용하여 관절의 3차원 좌표를 추정한다. 그 결과를 쿼터니언으로 변환하여 회전한 후, FBX 형식의 애니메이션을 생성한다. 제안한 방법의 정확도 측정을 위해 신체에 마커를 부착하여 마커의 3차원 위치를 바탕으로 생성한 애니메이션과 제안된 시스템으로 생성한 애니메이션의 오차를 비교하여 시스템 동작을 입증하였다.

Abstract

Recently, in various fields such as games, movies, and animation, content that uses motion capture to build body models and create characters to express in 3D space is increasing. Studies are underway to generate animations using RGB-D cameras to compensate for problems such as the cost of cinematography in how to place joints by attaching markers, but the problem of pose estimation accuracy or equipment cost still exists. Therefore, in this paper, we propose a system that inputs RGB images into a joint estimation network and converts the results into 3D data to create FBX format animations in order to reduce the equipment cost required for animation creation and increase joint estimation accuracy. First, the two-dimensional joint is estimated for the RGB image, and the three-dimensional coordinates of the joint are estimated using this value. The result is converted to a quaternion, rotated, and an animation in FBX format is created. To measure the accuracy of the proposed method, the system operation was verified by comparing the error between the animation generated based on the 3D position of the marker by attaching a marker to the body and the animation generated by the proposed system.

Keyword : Pose estimation, Quaternion, Joint rotation, FBX, 3D animation

Copyright ⓒ 2021 Korean Institute of Broadcast and Media Engineers. All rights reserved.
“This is an Open-Access article distributed under the terms of the Creative Commons BY-NC-ND (http://creativecommons.org/licenses/by-nc-nd/3.0) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited and not altered.”

Ⅰ. 서 론

최근 신체 모션 캡쳐를 이용하여 캐릭터를 생성하는 영화, 애니메이션, 게임 등 다양한 콘텐츠가 제공되고 있다. 이러한 콘텐츠를 제작할 때 이용하는 대부분의 모션 캡처 방식은 마커를 신체에 부착하고 마커 안의 센서를 통해 관절의 위치를 추정하는 방식이다. 하지만 이러한 방법은 촬영자에게 마커를 부착하는 것이 필수 불가결이고 촬영 장비에 대한 비용 문제가 있으며 마커의 특성으로 인해 촬영 장소에 제약이 있다. 이러한 제약을 줄이기 위해 마커를 신체에 부착할 필요 없이 RGB-D 카메라를 이용하여 신체 관절을 추정하고 신체 애니메이션을 생성하는 연구가 진행되고 있다[1,2]. 대표적인 RGB-D 카메라인 Microsoft에서 개발한 키넥트(Kinect)의 경우 깊이 센서를 이용하여 신체 맵을 형성하고, 머신러닝을 이용하여 관절의 위치를 추정한다. 그리고 추정된 관절 간의 방향을 이용하여 쿼터니언 형식으로 변환한 후 회전 값을 도출하는 Kinect Animation Studio[3]를 이용하여 3차원 데이터 형식인 FBX(Filmbox)으로 변환함으로써 애니메이션을 생성한다. 하지만 RGB-D 카메라를 활용하는 경우 마커 없이 RGB 카메라와 깊이 센서를 이용하여 관절을 추정하기 때문에 마커를 이용한 경우보다 정확도가 낮고, 장비 비용의 문제가 여전히 존재한다.

RGB-D 카메라를 활용하지 않고 RGB 카메라를 통해 2차원 신체 이미지를 획득하고, 딥러닝을 이용하여 관절을 추정함으로써 앞서 언급된 문제를 보완할 수 있다. 기존 RGB 이미지의 관절 추정 방법은 관절에 해당하는 이미지를 추출하여 HOG(Histogram of Oriented Gradient)와 같은 특징 추출 알고리즘으로 구성된 DPM(Deformable Part Model) 방식을 이용하였다[4,5]. 하지만 Tompson et al.[6]과 OpenPose[7], Martinez et al.[8]과 같이 관절 상호 관계성을 고려하는 딥러닝 모델을 통해 RGB 이미지의 관절 추정 정확도가 향상되었다. 따라서 딥러닝을 활용하여 관절을 추정한 후 3차원 데이터로 변환함으로써 RGB-D 카메라를 이용하여 애니메이션을 생성하는 방법보다 정확도를 보완하면서 애니메이션을 제작하는 비용을 절감할 수 있다. OpenMMD[9], AnimePose[10]는 RGB 이미지의 관절을 추정하여 애니메이션을 생성하는 시스템으로 OpenMMD는 독립적인 애니메이션 확장자로 출력되어 다양하게 활용할 수 없으며, 실시간으로 영상을 촬영하고 이를 애니메이션으로 변환하는데 어려움이 있다. AnimePose는 관절 추정 후 생성된 애니메이션을 Unity에 표출할 수 있지만, 애니메이션을 추출할 수 없으며, 실시간으로 영상을 촬영하고 애니메이션을 생성할 수 없다.

따라서 본 논문에서는 RGB 카메라를 통해 얻은 영상에서 딥러닝을 이용하여 관절을 추정하고, 이 추정된 관절을 FBX 데이터 형식으로 변환하여 애니메이션을 생성하는 것을 제안한다. 먼저 RGB 카메라로 신체 이미지를 획득하여 OpenPose를 통해 2차원 관절을 추정하고, FCRN[11]을 통해 이미지의 깊이를 추정한다. 이후 이 2차원 관절을 간단한 딥러닝 네트워크를 통해 3차원 관절로 변환한다. 그리고 3차원으로 이루어진 각 관절의 방향을 통해 쿼터니언(quaternion)으로 구성하여 회전시키고 추정된 깊이로 신체의 중심을 보정한다. 마지막으로 회전된 값을 FBX 형식으로 변환하여 최종 신체 애니메이션을 생성한다. 애니메이션 생성의 정확도 측정을 위해 신체에 마커를 부착하여 마커의 3차원 위치 정보를 바탕으로 생성한 애니메이션과 제안 시스템으로 생성한 애니메이션의 오차를 비교하였다. 평균 오차는 40.4mm로 팔과 다리 끝 관절이 고관절, 어깨, 목 관절보다 오차가 약 6배 컸다.

본 논문의 구성은 다음과 같다. Ⅱ장에서 2차원 관절의 위치와 깊이 추정 및 3차원 관절 위치 추정과 쿼터니언 회전 및 FBX를 이용한 애니메이션 생성 알고리즘에 대해 설명하고, Ⅲ장에서 제안하는 시스템의 구현 과정을 설명하였다. 그리고 Ⅳ장에서 제안한 방법에 대한 실험 환경을 설명하고 실험 결과에 대해 고찰하며, 마지막으로 Ⅴ장에서 결론을 맺는다.

a) 서울과학기술대학교 IT미디어공학과(Dept. of IT Media Engineering, The Graduate School, Seoul National University of Science and Technology)
b) 서울과학기술대학교 정보통신미디어공학전공(Dept. of Information Technology and Media Engineering, The graduate School of Nano IT Design Fusion, Seoul National University of Science and Technology)
c) 서울과학기술대학교 전자IT미디어공학과(Dept. of Electronic IT Media Engineering, Seoul National University of Science and Technology)
‡ Corresponding Author : 박구만(Gooman Park)
E-mail: gmpark@seoultech.ac.kr
Tel: +82-2-970-6425
ORCID: http://orcid.org/0000-0002-7055-5568
※ This work was supported by Institute for Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) in 2021(No. 2017-0-00217, Development of Immersive Signage Based on Variable Transparency and Multiple Layers).
· Manuscript received April 12, 2021; Revised May 14, 2021; Accepted May 14, 2021.

Ⅱ. 관련 기술

1. RGB 이미지의 2차원 관절 추정

신체 관절 추정 알고리즘은 크게 top-down 방식과 bottom-up 방식으로 구성되어 있는데, top-down 방식은 사람을 먼저 검출한 후에 각 사람의 관절을 추정하며, bottom-up 방식은 관절을 먼저 인식하고 관계를 연결하여 최종 신체 관절을 추정한다. OpenPose는 bottom-up 기반 알고리즘으로 관절을 연결하는 과정에서 나타나는 계산량 증가 등의 문제를 보완하여 기존 알고리즘보다 정확도가 높고 속도가 빠르다. OpenPose의 2차원 관절 추정 흐름도는 아래 그림 1과 같다. 먼저 RGB 이미지를 입력하여 두 가지 지표를 예측하는데, 첫 번째는 각 관절이 이미지의 어느 부분에 위치할 것이라는 지표인 confidence score를 히트맵(heatmap)으로 표현하여 예측한다. 두 번째는 각 관절 사이의 벡터 방향 정보를 나타내는 지표인 affinity fields를 예측한다. 이 두 가지 지표를 반복적으로 예측하여 관절의 위치와 어느 신체에 속하는지 추정한다. 본 연구에서는 실시간으로 RGB 신체 이미지의 관절을 추정하기 위해 OpenPose를 사용하였다.

2. RGB 이미지의 깊이 정보 추정

RGB 이미지를 이용하여 3차원 가상공간에 애니메이션을 생성할 때 이미지의 깊이 정보를 알 수 없어 3차원 공간이 추상적으로 구축될 수 있다. 따라서 신체를 자연스럽게 표출할 수 있도록 이미지의 깊이를 추정하여 조정할 필요가 있다. 깊이는 이미지를 촬영한 카메라의 시점에서 객체까지의 거리를 의미한다. 기존의 딥러닝을 이용한 깊이 추정 알고리즘은 컨볼루션 네트워크 이후에 후처리 단계로 랜덤 포레스트와 같은 주변 데이터를 고려하여 라벨을 예측하는 통계 모델인 CRF (Conditional Random Field) 기반 정규화와 결합된 상태가 많았다[12,13]. 하지만 이러한 방법은 컨볼루션 네트워크와 CRF의 공동 사용에 관련된 많은 매개변수로 인해 학습 과정이 복잡하다. 따라서 다른 후처리 알고리즘을 사용하지 않고 ResNet[14]과 업샘플링으로 구성된 FCRN을 이용하여 깊이를 추정하였다. 완전 컨볼루션 네트워크(FCN)를 사용하여 더 높은 해상도로 출력하고, 동시에 더 적은 매개변수를 이용하여 학습하기 때문에 속

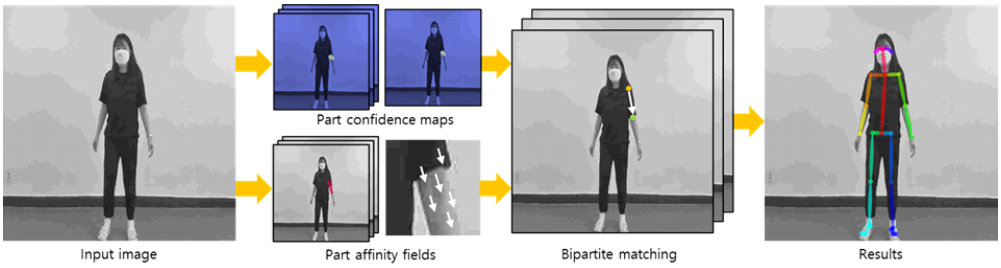


그림 1. 2차원 관절 추정 흐름도
Fig. 1. Flowchart of 2D Joint Estimation

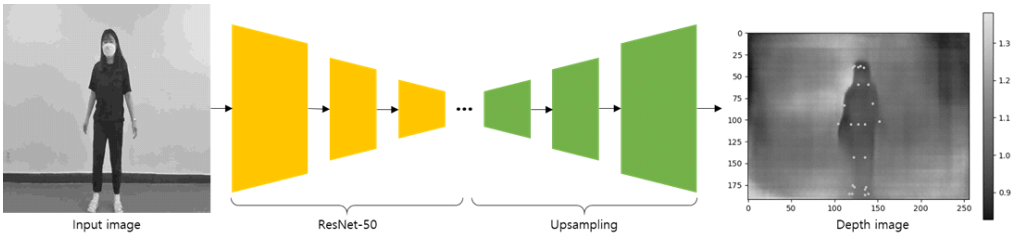


그림 2. FCRN를 이용한 깊이 추정 흐름도
Fig. 2. Flowchart of Depth Estimation Using FCRN

도가 빠르고 높은 성능을 보인다. 본 연구에서는 그림 2와 같이 FCRN을 이용하여 RGB 이미지의 깊이를 추정하고, 이 값을 통해 신체 중심을 보정하였다.

3. 2차원 관절 데이터의 3차원 변환

RGB 이미지에서 딥러닝을 이용하여 3차원 관절을 추정하는 방법은 두 가지로 구분하는데, 첫 번째는 이미지 전체를 입력하여 3차원 관절을 추정하는 방법이다. 두 번째 방법은 인조 데이터에 대한 학습을 통해 3차원 관절을 추정하는 방법으로, 실제 3차원 관절 데이터셋이 부족한 부분을 보완하기 위해 임의의 배경에 3차원 관절 정보가 입력된 사람 이미지를 합성하여 학습하였다. Martinez et al.은 두 번째 방법에 초점을 맞춰 이미지 전체를 보는 것이 아닌 관절 추정을 분리함으로써 접근하였는데, 즉 인조 데이터를 만들어서 학습하는 것이 아닌 사람의 관절 데이터만 추출하여 학습하였다. 따라서 네트워크가 단순하면서 속도는 빠르고 성능은 높일 수 있어 이 네트워크를 이용하였다. Martinez et al.는 2차원 관절 데이터를 3차원으로 변환하는 딥러닝 네트워크로 ResNet에서 제안했던 residual connection 구조를 이용한다. 하나의 선형 레이어를 선형 회귀 연산과 배치 정규화, ReLU, dropout으로 구성하였을 때, 두 개의 선형 레이어 결과와 입력값을 더함으로써 하나의 residual 블록을 형성한다. 제안 네트워크는 먼저 선형 레이어를 한번 진행한 후, residual 블록을 두 번 반복하고, 마지막으로 선형 회기 연산을 진행하여 3차원 데이터를 출력한다. 이 과정을 그림으로 나타내면 아래 그림 3과 같으며, 본 연구에서는 이 네트워크를 통해 2차원 관절 데이터를 3차원 관절 데이터로 변환하였다.

4. 3차원 데이터의 회전

3차원 관절 데이터를 이용하여 신체 애니메이션을 생성하기 위해서 관절 간의 회전 값을 계산하여 프레임이 지나갈수록 관절이 자연스럽게 움직일 수 있도록 해야 한다. 관절의 3차원 좌표를 회전시키기 위해 이용하는 방법은 두 가지가 있는데, 하나는 오일러 각이고, 나머지 하나는 쿼터니언을 이용하는 것이다. 오일러 각은 x축, y축, z축을 각각 따로 회전하는 방식을 가지는데, 각 회전축이 순차적으로 진행되는 중에 축이 겹쳤을 때 회전이 구분되지 않은 현상인 짐벌락이 발생할 수 있다. 이런 짐벌락 현상을 해결하기 위해 오일러 방법이 아닌 쿼터니언 방법을 이용한다. 쿼터니언은 3차원 공간에서 회전을 표현할 때 이용하는 개념으로 하나의 실수부(스칼라부)와 3개의 허수부(벡터부)로 구성된 복소수 체계이며 $q=(w, xi, yj, zk)$와 같이 표현할 수 있다. 입력된 3차원 벡터 v를 크기가 1인 단위 쿼터니언 q와 켤레 쿼터니언 $\bar{q}=(w, -xi, -yj, -zk)$를 양쪽에 곱한 $qv\bar{q}$를 통해 회전 변환하는데 이를 행렬로 변환하면 아래 식 1과 같다.

$$M_q = \begin{bmatrix} 1-2y^2-2z^2 & 2xy-2wz & 2xz+2wy \\ 2xy+2wz & 1-2x^2-2z^2 & 2yz-2wx \\ 2xz-2wy & 2yz+2wx & 1-2x^2-2y^2 \end{bmatrix} \quad (1)$$

본 연구에서는 2.3절에 언급한 네트워크를 이용하여 각 관절의 3차원 좌표를 계산하고 관절의 방향 벡터를 이용하여 회전 행렬로 변환하였다. 이 회전 행렬을 바탕으로 관절

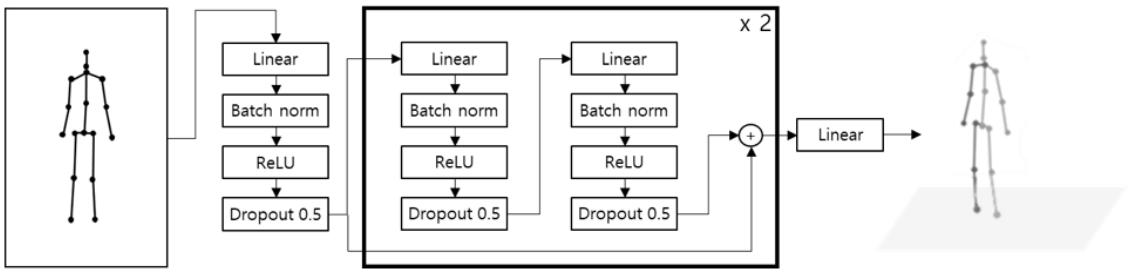


그림 3. 3차원 관절 데이터 생성 네트워크
Fig 3. 3D joint data generation network

의 쿼터니언을 구하고, 이후 부모 관절에 따라 자식 관절의 위치를 정하는 순운동학(Forward Kinematics: FK)을 이용하여 관절의 쿼터니언 회전을 계산하였다. 이때 최상위 부모 관절인 신체 중심(척추 부분)으로 각 관절의 회전을 계산하기 때문에 신체 중심은 회전값을 구하지 않고 3차원 관절 위치를 가상공간에 표현한다. 따라서 2.2절에 언급한 네트워크를 이용하여 추정된 깊이 정보를 바탕으로 신체 중심 위치를 보정한다.

5. FBX

FBX는 3차원 데이터를 다루기 위한 도구 집합으로 FBX SDK(Software Development Kit)[15]를 이용하여 직접 구현이 가능하며, FBX SDK의 구조는 그림 4와 같다. FBX SDK는 manager, scene, object, property, connection, node, attribute와 같은 개념으로 추상화되어 있는데, 먼저 FbxManager를 정의하고 FbxScene을 할당한다. 이후 이 FbxScene에 평행이동(translation), 회전(rotation), 스케일(scaling)과 같은 property가 connection된 object를 배치한다. 그리고 이 object들은 모두 계층 구조로 구성된 node로 표현되며 카메라, 빛과 같은 node에 대한 자료들은 attribute에 저장되어 있다. 이 node를 정의함으로써 신체 스켈레톤을 구성하고, 관절들의 부모-자식 관계를 정의하여 골격 형태를 만든다. 부모-자식 관계를 바탕으로 프레임 별 애니메이션을 생성하는데, 자식 관절의 회전 쿼터니언을 행렬로 변환한다. 그리고 이 행렬과 프레임 시간을 이용하여 커브 값을 계산하고 애니메이션을 생성한다. 본 연구에서는 FBX SDK를 이용하여 각 관절의 쿼터니언 데이터를 애니메이션으로 생성하였으며, 이를 3차원 가상공간에 표출하였다.

Ⅲ. 시스템 구현

1. 시스템 개요

본 논문에서 제안하는 시스템은 딥러닝 기반 관절 추정 네트워크를 이용하여 RGB 이미지의 관절을 추정하고 3차원 데이터로 변환하여 애니메이션을 생성하는 시스템으로 전체 구성도는 그림 5와 같다. 먼저 RGB 이미지를 OpenPose에 입력하여 2차원 관절을 추정하고, FCRN을 이용하여 깊이를 추정한다. 그리고 2차원 관절 정보를 residual 구조로 이루어진 간단한 딥러닝 네트워크에 입력하여 3차원 관절 값을 계산한다. 이후 이 값의 방향 벡터를 이용하여 회전 행렬을 계산하고, 관절의 쿼터니언을 구성한 후 순운동학 기반 쿼터니언 회전 연산한다. 신체 중심의 경우 추정된 깊이 정보를 바탕으로 위치를 보정한다. FBX에 애니메이션을 생성하기 위해 관절 노드 정보가 정의된 FbxScene을 설정하고, 각 관절의 회전 쿼터니언을 FbxScene에 적용함으로써 애니메이션을 생성한다. 마지막으로 이 값을 저장 및 가상공간에 표출한다.

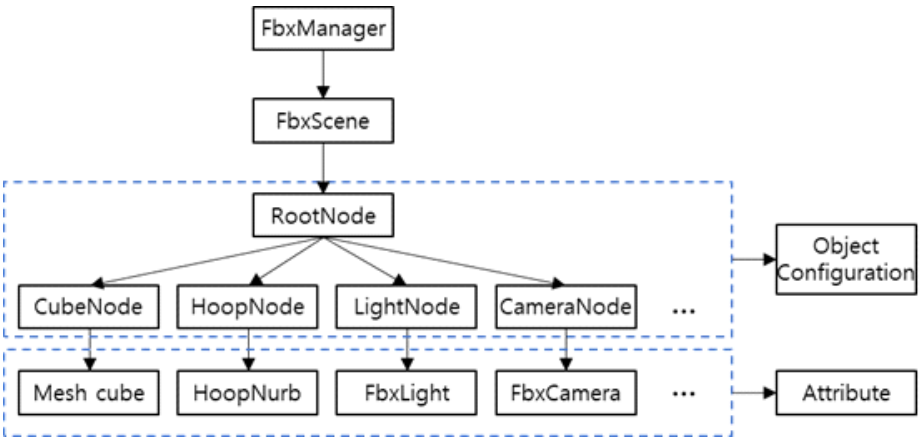


그림 4. FBX SDK 구조
Fig. 4. FBX SDK Structure

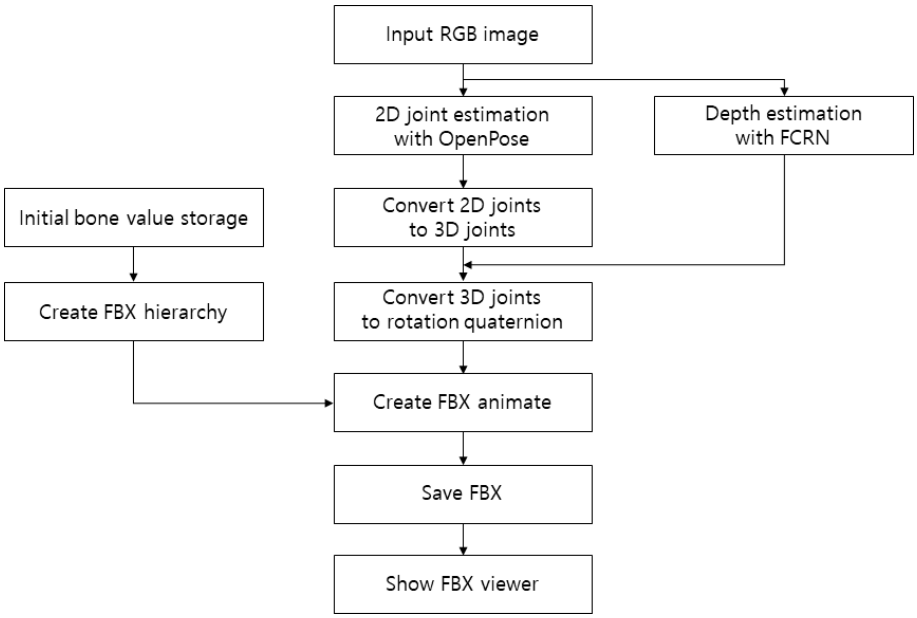


그림 5. 제안 시스템 구조
Fig. 5. Proposal system structure

2. 2차원 및 3차원 관절 추정

2차원 및 3차원 관절을 추정하기 위한 알고리즘 구성도는 그림 6과 같다. 먼저 RGB 이미지를 신체 애니메이션으로 생성하기 위해서 입력된 이미지의 관절 좌표를 추정해야 한다. 2차원 관절 추정은 2차원 관절 데이터셋인 MS COCO[16], MPII[17], BODY_25, AI Challenger[18] 중 3차원 애니메이션을 만들었을 때 오차가 적고 적당한 관절 개수를 가진 BODY_25 데이터셋으로 학습된 OpenPose 네트워크를 이용하였다[19,20]. 이후 2차원 관절 좌표를 3차원 좌표로 변환하기 위한 네트워크에 입력하는데, 관련 기술에 언급하였듯이 선형 레이어를 한번 진행한 후 2개의 선형 레이어로 구성된 residual 블록을 두 번 진행하고 선형 회귀를 연산하여 3차원 좌표를 얻는다. 이 네트워크의 경우 Human3.6M[21] 데이터셋으로 학습하였기 때문에 입력이 되는 2차원 좌표를 Human3.6M 데이터셋 형식에 맞게 변환해야 한다. 따라서 25개 관절 좌표 중 고관절 중심과 양쪽 눈, 발가락 및 발꿈치 좌표를 제외한 나머지 좌표만 추출한다. 그리고 코, 어깨 중심, 척추와 같이 Human3.6M 형식에 있지만, BODY_25 형식에 없거나 보정이 필요한 관절은 인접한 관절을 이용하여 위치를 조정한다. 다음 표 1은 BODY_25와 Human3.6M 데이터셋의 관절 번호와 계산식이며, 이때 R

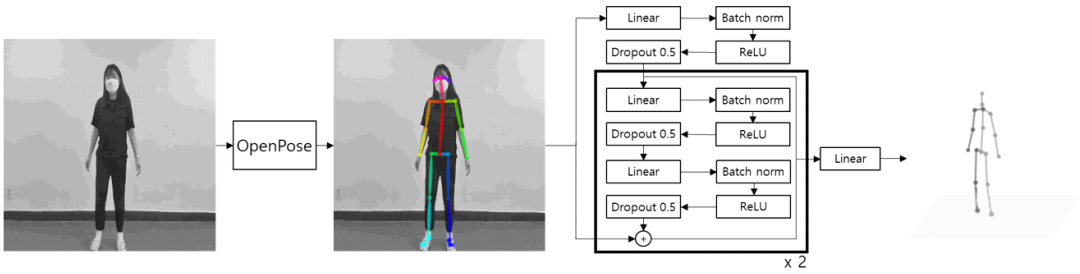


그림 6. 2차원 및 3차원 관절 추정 구성도
Fig. 6. 2D and 3D joint estimation diagram

표 1. BODY_25와 Human3.6M 데이터셋의 관절 번호와 계산식
Table 1. Joint number and formula for BODY_25 and Human3.6M datasets

BODY_25 index	Name	Human3.6M index	Name	Calculation
0	Nose	0	Mid Hip	(R Hip + L Hip) / 2
1	Neck	1	R Hip	Same as BODY_25
2	R Shoulder	2	R Knee	Same as BODY_25
3	R Elbow	3	R Ankle	Same as BODY_25
4	R Wrist	4	L Hip	Same as BODY_25
5	L Shoulder	5	L Knee	Same as BODY_25
6	L Elbow	6	L Ankle	Same as BODY_25
7	L Wrist	7	Spine	(Mid Hip + Thorax) / 2
8	Mid Hip	8	Thorax	Neck*1.1 + Mid Hip*0.1
9	R Hip	9	Neck/Nose	(Nose + Thorax) / 2
10	R Knee	10	Head	Nose
11	R Ankle	11	L Shoulder	Same as BODY_25
12	L Hip	12	L Elbow	Same as BODY_25
13	L Knee	13	L wrist	Same as BODY_25
14	L Ankle	14	R Shoulder	Same as BODY_25
15	R Eye	15	R Elbow	Same as BODY_25
16	L Eye	16	R wrist	Same as BODY_25
17	R Ear	-	-	-
18	L Ear	-	-	-
19	L BigToe	-	-	-
20	L SmallToe	-	-	-
21	L Heel	-	-	-
22	R BigToe	-	-	-
23	R SmallToe	-	-	-
24	R Heel	-	-	-

은 오른쪽(Right), L은 왼쪽(Left)을 의미한다.

보정된 관절의 2차원 좌표를 정규화한 후 3차원 관절 추정 네트워크에 입력하면 두 개의 residual block을 거쳐 정규화된 3차원 좌표를 출력한다. 후에 가상환경에서 신체 관절을 표출할 때 카메라가 바라보는 방향에 관절을 생성하도록 각 관절이 카메라 좌표계를 따르도록 변환한다. 카메라 좌표계는 카메라가 기준이 되는 좌표계로 카메라 렌즈가 바라보고 있는 방향을 Z축, 카메라부터 바닥까지의 방향을 Y축, 카메라의 옆 방향을 X축이 되도록 3차원 위치를 변경한다. 이 과정에서 OpenPose 결과와 비교하고 골격의 길이나 프레임 전후의 이동 평균을 계산함으로써 관절의 위치를 더욱 안정화하였다. 이후 서 있을 때 발목의 위치를 바닥으로 이동하기 위해 모든 프레임의 발목 위치를 통해 발목 중앙값을 계산하고, 모든 관절의 y 좌표를 발목의 중앙값만큼 평행이동하였다.

3. 쿼터니언 회전 변환

신체 애니메이션을 생성하기 위해 3.2절의 결과인 3차원 관절 좌표를 이용하여 각 관절의 쿼터니언 회전을 계산한다. 앞서 언급하였듯이 애니메이션을 만들기 전 관절 노드의 부모-자식 관계를 정의하였는데, 그 관계에 맞게 쿼터니언 구성 후 회전해야 한다. 애니메이션에 표현되는 최종 관절 구조와 각 관절의 부모-자식 관계는 그림 7과 같으며 화살표가 나가는 방향은 부모 노드, 화살표가 가리키는 방향은 자식 노드가 된다. 즉, 최상위 부모 관절은 센터(center)로 그 아래 상반신(upper body)과 하반신(lower body)이 자식 노드로 정의된다. 최상위 노드인 센터를 기준으로 신체 관절이 구성되기 때문에 센터는 쿼터니언 회전 값이 아닌 위치(x, y, z) 값으로 구성된다. 3.2절의 3차원 좌표는 추상적으로 예측된 값이기 때문에 FCRN을 통해 추정된 깊이를

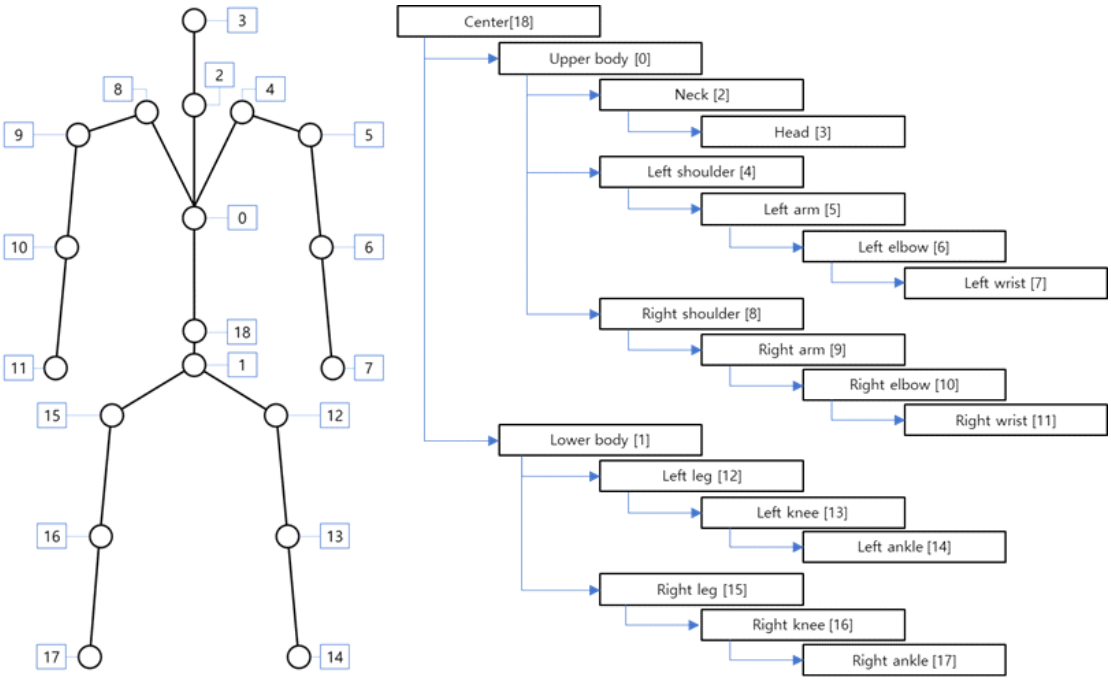


그림 7. 애니메이션에 표출되는 최종 관절 구조(좌)와 부모-자식 관계(우) 정의
Fig. 7. Defining the final joint structure (left) and parent-child relationship (right) displayed in the animation

이용하여 센터의 위치를 조정한다. 식 2와 같이 목(neck)과 왼쪽 고관절(left hip), 오른쪽 고관절(right hip)의 깊이 평균을 계산하여 이 값을 센터의 z 좌표에 입력하였다.

$$Center.z = (depth_{Neck} + depth_{Left\,Hip} + depth_{Right\,Hip}) / 3 \quad (2)$$

이후 순운동학 연산을 기반으로 쿼터니언 회전을 계산하는데, 순운동학 연산은 부모 관절에 따라 자식 관절의 위치를 변화시키기 때문에 자식 관절이 없는 머리(head), 양쪽 손목(left wrist, right wrist)과 양쪽 발목(left ankle, right ankle)은 쿼터니언 회전을 계산하지 않는다. 따라서 이 관절들과 센터를 제외한 관절들의 쿼터니언 회전을 계산하기 위해 먼저 3.2절의 3차원 좌표를 이용하여 각 관절의 방향을 쿼터니언으로 나타내야 한다. 해당 관절의 인접한 관절을 이용하여 방향 벡터를 구하고, 이 벡터를 행렬로 표현한 후 행렬의 인자를 이용하여 쿼터니언을 구성할 수 있다. 즉, 구하고자 하는 관절의 방향 벡터를 direction과 up이라고 하였을 때, direction과 up을 외적 연산한 결과까지 총 세 가지의 방향 벡터를 구성할 수 있다. 상반신의 경우를 예로 들어 direction은 어깨 중심(표 1의 우측 8번 관절)에서 척추(표 1의 우측 7번 관절)를 뺀 방향 벡터이고, up은 direction과 오른쪽 어깨(표 1의 우측 14번 관절)에서 왼쪽 어깨(표 1의 우측 11번 관절)를 뺀 방향 벡터를 외적 연산한 결과이다.

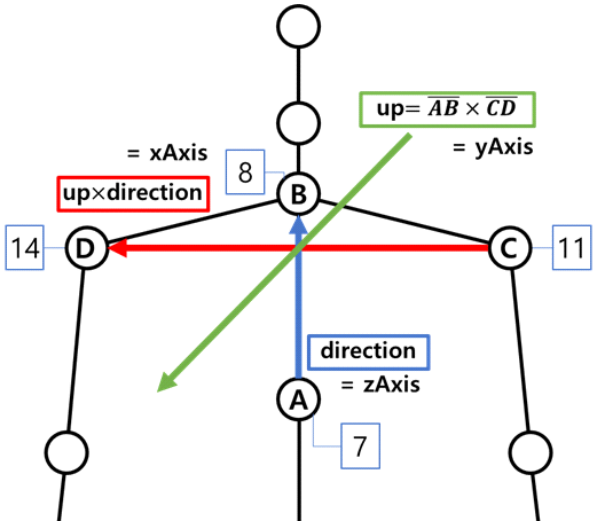


그림 8. 상반신의 방향 벡터 구성
Fig. 8. Composition of the direction vector of the upper body

이 direction과 up을 외적 연산하여 새로운 방향 벡터를 계산하고 이를 정규화하여 xAxis라고 정의한다. 이어서 up을 정규화하여 yAxis라 정의하고, direction을 정규화하여 zAxis라 정의한다. 그림 8은 3차원 관절 구성도의 일부로, 위에 언급한 상반신의 세 가지 방향 벡터 direction (zAxis), up(yAxis), direction×up(xAxis)을 나타내며, 세 벡터는 서로 수직을 이룬다.

세 가지 방향 벡터 xAxis, yAxis, zAxis를 이용하여 3x3 회전 행렬로 나타낼 수 있는데, 이는 식 3과 같다. 이 회전 행렬은 2.4절에서 언급한 식 1과 동일하므로 각 인자를 통해 1개의 스칼라 w와 3개의 벡터 x, y, z로 이루어진 쿼터니언을 구성할 수 있으며 이는 아래 식 4와 같다. 이때 $M_q(0,0)+M_q(1,1)+M_q(2,2)$은 0보다 크다.

$$M_q=\begin{bmatrix}\text{xAxis}.x & \text{yAxis}.x & \text{zAxis}.x\\ \text{xAxis}.y & \text{yAxis}.y & \text{zAxis}.y\\ \text{xAxis}.z & \text{yAxis}.z & \text{zAxis}.z\end{bmatrix} \tag{3}$$

정리하면 인접한 관절의 3차원 위치 값을 이용하여 방향 벡터 direction, up, direction×up을 계산한 후 이 세 가지 벡터를 3x3 회전 행렬로 나타내어 쿼터니언으로 구성할 수 있다. 아래 표 2는 각 관절을 쿼터니언으로 구성하기 위한 direction과 up의 유도 식이다. 식은 표 1의 Human3.6M 관절 이름을 바탕으로 작성하였다.

각 관절의 direction과 up 방향 벡터를 이용하여 쿼터니언으로 나타낸 후 식 5와 같이 순운동학 기반 관절의 쿼터니언 회전을 계산하였다. P'는 구하고자 하는 관절의 쿼터니언 회전 값이고, n은 센터를 제외한 부모 관절의 개수이다. B_i는 i번째 부모 관절의 쿼터니언 회전 값이고, P는 현재 관절의 쿼터니언 구성이며, w는 보정용 가중치로 단위 쿼터니언을 입력하였다. 순운동학 연산은 부모 관절의 회전에 따라 자식 관절을 변화시키기 때문에 해당 관절의 센터를 제외한 모든 부모 관절의 회전 쿼터니언을 적용하였다. 예를 들어 왼쪽 무릎(left knee)의 회전 쿼터니언을 구할 때, B에 입력되는 값은 왼쪽 무릎의 부모 관절인 왼쪽 다리(left leg)와 하반신(lower body)의 회전 쿼터니언으로 이 쿼터니언의 역수를 모두 곱한다. 이후 왼쪽 무릎의 회전 값을 곱하고 보정용 가중치를 곱하여 왼쪽 무릎의 회전 쿼터니언을 계산할 수 있다.

$$\begin{aligned}
w &= 0.25 * 2 * \sqrt{M_q(0,0)+M_q(1,1)+M_q(2,2)+1}\\
x &= (M_q(2,1)-M_q(1,2))/2 * \sqrt{M_q(0,0)+M_q(1,1)+M_q(2,2)+1}\\
y &= (M_q(2,0)-M_q(0,2))/2 * \sqrt{M_q(0,0)+M_q(1,1)+M_q(2,2)+1}\\
z &= (M_q(0,1)-M_q(1,0))/2 * \sqrt{M_q(0,0)+M_q(1,1)+M_q(2,2)+1}
\end{aligned} \tag{4}$$

표 2. 각 관절의 direction과 up 유도
Table 2. Induction of direction and up of each joint

Joint name	Direction	Up
Upper body	Thorax - Spine	Direction × (R shoulder - L shoulder)
Lower body	Mid hip - Spine	Direction × (L hip - R hip)
Neck	Neck - Thorax	(R shoulder - L shoulder) × Direction
Left shoulder	L shoulder - Thorax	Direction × (R shoulder - L shoulder)
Left arm	L elbow - L shoulder	Direction × L wrist - L elbow)
Left elbow	L wrist - L elbow	(L elbow - L shoulder) × Direction
Right shoulder	R shoulder - Thorax	Direction × (R shoulder - L shoulder)
Right arm	R elbow - R shoulder	Direction × R wrist - R elbow)
Right elbow	R wrist - R elbow	(R elbow - R shoulder) × Direction
Left leg	L knee - Mid hip	Direction × (L ankle - L knee)
Left knee	L ankle - L Knee	(L knee - Mid hip) × Direction
Right leg	R knee - Mid hip	Direction × (R ankle - R knee)
Right knee	R ankle - R Knee	(R knee - Mid hip) × Direction

$$P' = w\prod_{i=0}^{n}(B_i^{-1})P \quad (5)$$

4. FBX 형식 애니메이션 생성

관절의 쿼터니언 회전 값과 센터의 위치를 이용하여 FBX 형식 애니메이션을 생성하기 위해 먼저 FbxManager를 정의하고 FbxScene을 할당한다. 이후 이 FbxScene에 각 관절에 대한 Node를 정의하고 그림 7에 나타낸 바와 같이 부모-자식 관계를 지정하며 관절의 위치, 크기, 회전 정보를 설정한다. 이때 설정하는 관절 정보는 가상환경에 표출될 신체 모델에 대한 관절 간 길이와 관절 위치를 의미하며 할당한 값은 표 3과 같다. 그림 9에서도 알 수 있듯이, y는 가상공간의 바닥과 수직이며 |y|는 관절 간의 길이를 의미한다. x와 z 역시 x축, z축 방향으로 설정된 값 만큼 멀어지는 것을 의미하는데, 멀어지는 기준은 가상환경의 원점이 아닌 부모 관절이다.

표 3. 각 관절의 초기 위치 및 길이 설정
Table 3. Set the initial position and length of each joint

No.	Joint name	x	y	z
0	Upper body	0	13.3	-0.26
1	Lower body	0	-5	-0.26
2	Neck	0	16.53	0
3	Head	0	16.9	0
4	Left shoulder	0.14	16.13	0.26
5	Left arm	1.15	15.69	0.32
6	Left elbow	3.46	14.28	0.48
7	Left wrist	5.02	12.96	0.25
8	Right shoulder	-0.14	16.13	0.26
9	Right arm	-1.15	15.69	0.32
10	Right elbow	-3.46	14.28	0.48
11	Right wrist	-5.02	12.96	0.25
12	Left leg	10.8	-5.75	-0.07
13	Left knee	0.67	26.09	-0.05
14	Left ankle	0.88	21.34	0
15	Right leg	-10.8	-5.75	-0.07
16	Right knee	-0.67	26.09	-0.05
17	Right ankle	-0.88	21.34	0
18	Center	0	56	0

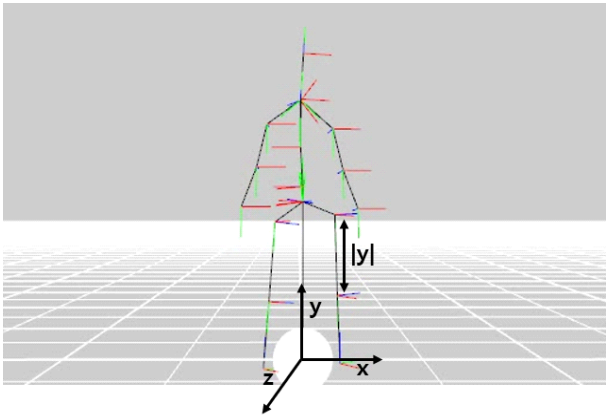


그림 9. 신체 모델 초기 설정을 위한 x, y, z
Fig. 9. x, y, z for initial body model setup

이후 프레임 전환 시간을 설정하여 애니메이션 속도를 조정한다. FbxScene에 애니메이션 데이터 저장소인 FbxAnimStack을 정의하는데, FbxAnimStack은 애니메이션 커브 값이 저장된 FbxAnimLayer를 담는 역할을 한다. 즉 프레임마다 각 관절에 대한 애니메이션 커브 값을 계산하여 FbxAnimStack에 담고, 이 정보들을 FBX 파일로 저장한다. 각 관절의 애니메이션 커브 값을 얻기 위해 먼저 3.3절에서 계산한 쿼터니언 회전 값을 행렬로 변환하고 x, y, z 회전 값을 추출한다. 이후 각 노드의 회전 값과 앞서 언급한 프레임 전환 시간을 통해 베지에 곡선(Bezier Curves)를 구현함으로써 애니메이션을 생성한다. 이때 입력된 관절의 부모-자식 관계에 따라 계산 방식이 다른데, 관절의 자식 노드가 1개일 때와 1개가 아닐 때로 나뉜다. 입력된 관절의 자식 노드가 1개인 그림 7의 목(neck), 양쪽 어깨, 팔, 팔꿈치(shoulder, arm, elbow)는 자식 노드의 회전 쿼터니언을 입력하여 애니메이션을 생성한다. 하지만 자식 노드가 회전 쿼터니언 값이 존재하지 않는 머리(head), 양쪽 손목, 발목(wrist, ankle)일 경우 3차원 위치 좌표를 이용하여 각을 추정한다. 입력된 관절의 자식 노드가 1개가 아닌 상반신(upper body)과 하반신(lower body), 즉 자식 노드가 여러 개 존재할 때는 입력 관절의 회전 쿼터니언을 입력하여 애니메이션을 생성한다. 최상위 관절 센터 노드는 위치 좌표를 이용하며 이 과정을 모든 프레임에 반복한다. 이후 FBX 파일 형식으로 저장하고, 저장된 FBX 파일을 불러와서 설정된 가상공간에 애니메이션을 재생한다.

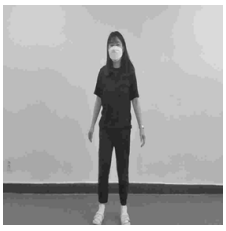
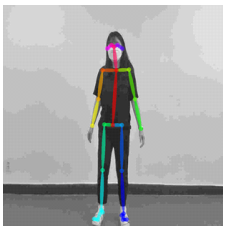
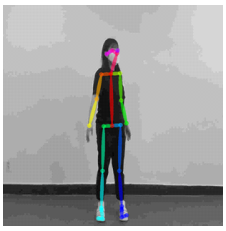
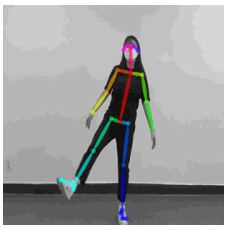
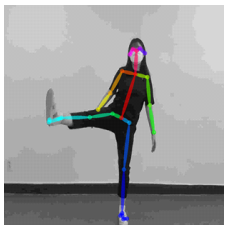
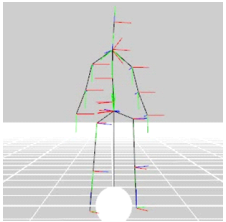
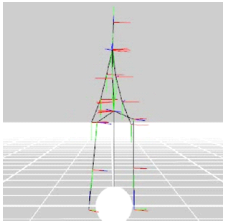
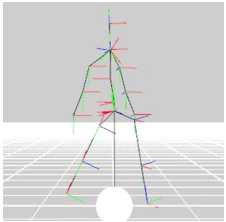
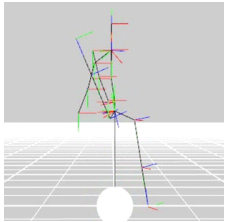
Ⅳ. 실 험

본 연구에서 RGB 신체 이미지의 2차원 관절 추정은 C++ Caffe 기반으로 구현하고, 3차원 관절 학습 및 추정은 Python의 tensorflow 1.10버전을 사용하였으며 회전 쿼터니언 구성은 Python의 PyQT5 QQuaternion 메소드를 사용하였다. 그리고 애니메이션 생성 및 가상공간 표출은 Open-Frameworks 0.9.8 환경에서 FBX SDK 2019를 이용하여 구현하였다.

본 실험은 딥러닝을 이용하여 RGB 카메라로 촬영한 데이터의 관절을 추정하고, 이 추정된 관절 값을 FBX 데이터 형식으로 변환하여 가상환경에 애니메이션이 생성되는지 확인하기 위해 진행되었다. 그리고 마커와 키넥트를 활용하여 추출한 실제 위치와 딥러닝을 활용하여 추정된 위치를 바탕으로 생성된 애니메이션의 오차를 비교하였다.

먼저 한 명이 실험에 참가하여 웹캠으로 다리 올리기, 상반신을 좌우로 돌리기 등 서 있는 자세 중심의 신체 동작을 촬영한 후 저장된 1,310개의 프레임에 대한 애니메이션을 생성하였다. 표 4는 애니메이션 생성 과정의 결과 이미지로 첫번째 행은 입력된 이미지이고, 두번째와 세번째 행은 2차원 및 3차원 관절 추정 결과이며, 마지막으로 네번째 행은 가상공간에 생성된 애니메이션 이미지이다. 애니메이션 생성 결과 바르게 서 있는 상태에서 상반신을 좌우로 돌린 경우와 같이 단순한 동작들은 제대로 재생된 것을 볼 수

표 4. 입력된 RGB 신체 이미지와 추정된 2차원 및 3차원 관절 이미지, 최종 애니메이션 이미지
Table 4. Input RGB body images, estimated 2D and 3D joint images, and final animation images

Input RGB images				
2D pose estimation				
3D pose estimation				
Animation images				

있다. 하지만 다리를 높게 올리는 것과 같이 몸을 크게 움직인 경우에는 추정된 관절과 일치하지 않고, 보다 큰 동작을 수행하거나 관절이 꼬인 상태로 애니메이션이 생성되었다.

비교를 위해 신체 관절 부위에 마커를 부착하고 키넥트를 통해 추출한 마커의 3차원 위치 좌표를 바탕으로 생성한 애니메이션과 제안 시스템을 이용하여 생성된 애니메이션의 오차를 비교하였다. 오차는 마커를 이용해 생성된 애니메이션과 제안 시스템에서 생성된 애니메이션과의 길이 차이를 비교하여 계산한다. 이때 키넥트로 더 정확한 3차원 좌표를 추정하기 위해 닫힘 연산과 중간 값 필터를 이용하여 생성된 깊이맵의 잡음을 줄였다[2]. 그림 10은 상반신을 좌우로 완전히 돌린 경우와 팔, 다리를 올리는 등 동작을 크게한 프레임에서 각 관절의 오차를 비교한 그래프로 그림 7의 관절 이름을 바탕으로 작성하였다. 머리(head), 목(neck), 센터(center), 상반신(upper body) 및 하반신(lower body)의 경우 서 있는 자세를 중심으로 실험하여 오차가 약 3.7mm로 오차가 작았다. 양쪽 어깨(left shoulder, right shoulder)의 경우 3차원 관절을 바탕으로 쿼터니언을 구성하였기 때문에 애니메이션에서 목과 위치가 일치하도록 생성되어 약간의 오차가 발생했다. 팔과 다리의 시작 부분(left arm, right arm, left leg, right leg)은 상반신을 좌우로 완전히 돌릴 때 오차가 약간 발생하였다. 그리고 나머지 팔꿈치와 손목, 무릎과 발목의 경우 크게 동작하거나 관절 간의 겹침이 있을 때 다른 관절과 비교하여 오차가 약 6배 크게 발생하였다. 딥러닝을 이용하여 RGB 영상의 관절을 추정하고, 이를 3차원 데이터로 변환하여 생성된 애니메이션을 가상환경에서 재생하였을 때 대부분 관절들의 오차가 작았다. 하지만 팔과 다리 시작 부분에서 손목과 발목으로 갈수록 점점 오차가 커졌는데, 이는 자식 관절이 부모 관절보다 위치 및 각도 변화가 크기 때문에 동작을 크게할수록 오차가 컸다.

V. 결 론

최근 게임, 영화, 애니메이션 다양한 분야에서 모션 캡처를 이용하여 신체 모델을 구축하고 캐릭터를 생성하여 3차원 공간에 표출하는 콘텐츠가 증가하고 있다. 마커를 부착하여 관절의 위치를 측정하는 방법에서 촬영 장비에 대한 비용과 같은 문제를 보완하기 위해 RGB-D 카메라를 이용하여 애니메이션을 생성하는 연구가 진행되고 있지만, 관절 추정 정확도나 장비 비용의 문제가 여전히 존재한다. 본 논문에서는 애니메이션 생성에 필요한 장비 비용을 줄이고 관절 추정 정확도를 높이기 위해 RGB 카메라를 통해 얻은 영상에서 딥러닝을 이용하여 관절을 추정하고, 이 추정된 관절 값을 FBX 데이터 형식으로 변환하여 애니메이션을

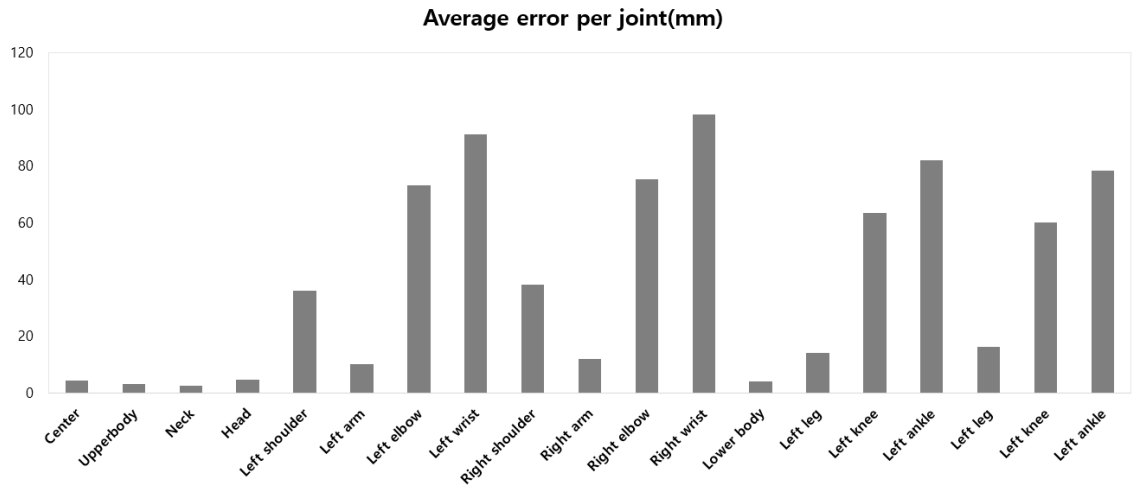


그림 10. 관절 별 평균 오차(mm)
Fig. 10. Average error per joint(mm)

생성하였다. 먼저 RGB 이미지에 대한 2차원 관절과 깊이를 추정하고, 2차원 관절을 간단한 딥러닝 네트워크에 입력하여 3차원 관절 값으로 변환하였다. 그리고 3차원으로 이루어진 각 관절 값의 방향을 통해 쿼터니언으로 구성하고, 추정된 깊이로 신체의 중심을 보정하였다. 마지막으로 관절 간의 관계를 바탕으로 회전시킨 후 이 값을 FBX 형식으로 변환하여 최종 신체 애니메이션을 생성하였다. 신체 관절에 마커를 부착하여 측정한 마커의 3차원 위치 정보를 바탕으로 생성한 애니메이션과 제안 시스템으로 생성한 애니메이션의 오차를 비교한 결과, 몸을 크게 움직이거나 관절 간의 겹침이 있는 경우에는 애니메이션이 추정된 관절과 일치하도록 생성되지 않았다. 센터, 상반신, 하반신, 목, 머리, 어깨, 고관절과 같이 신체 중심 축에 가까운 관절은 애니메이션이 대부분 올바르게 생성되었지만 팔꿈치와 손목, 무릎과 발목의 경우 크게 동작하거나 관절 간의 겹침이 있을 때 오차가 약 6배 크게 발생하였다. 향후 연구로 관절 추정과 회전 쿼터니언 구성, FBX 형식 애니메이션 생성까지의 과정을 하나의 프로세스로 구축하여 알고리즘을 간략화하고, 최근의 관절 추정 알고리즘을 분석하여 관절 추정 정확도를 높일 예정이다. 또한 애니메이션 생성 시 더 정확하게 관절이 회전할 수 있도록 커브 알고리즘을 보완하고 mesh 기반 3차원 오브젝트 모델링을 통해 더 정확한 비교 및 분석을 할 예정이다.

참 고 문 헌 (References)

[1] S. Kim, “Realtime 3D Human Full-Body Convergence Motion Capture using a Kinect Sensor,” Journal of Digital Convergence, Vol.14, No.1, pp.189-194, Jan 2016, https://doi.org/10.14400/JDC.2016.14.1.189

[2] J. Jeong, M. Yoon, S. Kim, and G. Park, “Design and production of real-time 3D animation viewer engine based on motion capture,” The Institute of Electronics and Information Engineers, 531-535, Jun 2019.

[3] Kinect animation studio, http://marcojrfurtado.github.io/KinectAnimationStudio/index.html

[4] Y. Yang and D. Ramanan, "Articulated Human Detection with Flexible Mixtures of Parts," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.35, No.12, pp.2878-2890, Dec 2013

[5] B. Sapp and B. Taskar, “MODEC: Multimodal decomposable models for human pose estimation,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp.3674 – 3681, 2013.

[6] J. Tompson, A. Jain, Y. LeCun, and C. Bregler, “Joint training of a convolutional network and a graphical model for human pose estimation,” Adv. Neural Inf. Process. Syst., Vol.2, pp.1799 – 1807, Jan 2014.

[7] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh, “Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields,“ Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7291-7299, 2017.

[8] J. Martinez, R. Hossain, J. Romero, and J. J. Little, “A Simple Yet Effective Baseline for 3d Human Pose Estimation,” Proc. IEEE Int. Conf. Comput. Vis., vol. 2017-October, pp.2659 – 2668, 2017.

[9] OpenMMD, https://github.com/peterljq/OpenMMD

[10] Kumarapu, Laxman and Prerana Mukherjee. “AnimePose: Multi-person 3D pose estimation and animation.” Pattern Recognit. Lett. 147, pp.16-24. 2021.

[11] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. “Deeper depth prediction with fully convolutional residual networks,” In 3D Vision (3DV), 2016 Fourth International Conference on, pp.239 – 248. IEEE, 2016.

[12] Bo Li, Chunhua Shen, Yuchao Dai, A. van den Hengel and Mingyi He, "Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.1119-1127, 2015, doi: 10.1109/CVPR.2015.7298715.

[13] Liu, Fayao, Chunhua Shen and Guosheng Lin. “Deep convolutional neural fields for depth estimation from a single image.” 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.5162-5170, 2015.

[14] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.770-778, 2016. doi: 10.1109/CVPR.2016.90.

[15] FBX SDK, https://www.autodesk.com/developer-network/platform-technologies/fbx-sdk-2020-0

[16] MS COCO dataset, https://cocodataset.org/#home

[17] MPII dataset, http://human-pose.mpi-inf.mpg.de/

[18] AI Challenger dataset, http://dataju.cn/Dataju/web/datasetInstanceDetail/440

[19] M. Yoon, Research of FBX generation using deep learning, Master’s Thesis of Seoul National University of Science and Technology, Seoul, Korea, 2020.

[20] S. Kim, Y. Lee, and G. Park. “Real-Time Joint Animation Production and Expression System using Deep Learning Model and Kinect Camera.“ Journal of Broadcast Engineering 26(3), pp.269-282, May 2021.

[21] Human3.6M dataset, http://vision.imar.ro/human3.6m/description.php

저 자 소 개

이 유 진

- 2014년 3월 ~ 2018년 2월 : 인천대학교 정보통신공학과 (공학사)
- 2020년 3월 ~ 현재 : 서울과학기술대학교 IT미디어공학과 석사과정
- ORCID : https://orcid.org/0000-0001-8342-0235
- 주관심분야 : 인공지능, 영상 처리, VR/AR

김 상 준

- 2017년 3월 ~ 2019년 2월 : 서울미디어대학원대학교 미디어공학전공 (공학석사)
- 2019년 3월 ~ 현재 : 서울과학기술대학교 정보통신미디어공학 박사과정
- ORCID : https://orcid.org/0000-0001-7498-6149
- 주관심분야 : 컴퓨터그래픽스, 증강현실, 프로젝션맵핑

박 구 만

- 1984년 : 한국항공대학교 전자공학과 공학사
- 1986년 : 연세대학교 대학원 전자공학과 석사
- 1991년 : 연세대학교 대학원 전자공학과 박사
- 1991년 ~ 1996년 : 삼성전자 신호처리연구소 선임연구원
- 1999년 ~ 현재 : 서울과학기술대학교 전자IT미디어공학과 교수
- 2006년 ~ 2007년 : Georgia Institute of Technology, Dept.of ECE. Visiting Scholar
- 2016년 ~ 2017년 : 서울과학기술대학교 나노IT디자인융합대학원 원장
- ORCID : https://orcid.org/0000-0002-7055-5568
- 주관심분야 : 컴퓨터비전, 실감미디어