특집논문 (Special Paper)
방송공학회논문지 제26권 제3호, 2021년 5월 (JBE Vol. 26, No. 3, May 2021)
https://doi.org/10.5909/JBE.2021.26.3.269
ISSN 2287-9137 (Online) ISSN 1226-7953 (Print)

딥러닝 모델과 Kinect 카메라를 이용한 실시간 관절 애니메이션 제작 및 표출 시스템 구축에 관한 연구

김 상 준[a], 이 유 진[b], 박 구 만[b)‡]

Real-Time Joint Animation Production and Expression System using Deep Learning Model and Kinect Camera

Sang-Joon Kim[a], Yu-Jin Lee[b], and Goo-man Park[b)‡]

요 약

증강현실과 가상현실 같은 3차원 콘텐츠 보급이 증가함에 따라 실시간 컴퓨터 애니메이션 기술의 중요성이 높아지고 있다. 하지만 컴퓨터 애니메이션 제작 과정은 대부분 수작업 혹은 마커를 부착하는 모션캡쳐 방식으로 이루어져 있다. 때문에 사실적인 영상을 얻기 위해서는 숙련된 전문가에게도 매우 오랜 시간이 필요하다. 이러한 문제점을 해결하기 위해 최근에는 딥러닝 모델과 센서를 기반으로 하는 애니메이션 제작 시스템과 알고리즘이 나오고 있다. 이에 본 논문에서는 딥러닝과 Kinect 카메라 기반 FBX 형식의 애니메이션 제작 시스템에서 자연스러운 인체 움직임을 구현하는 4가지 방법에 대해 연구했다. 각 방법은 환경적 특성과 정확도를 고려하여 선택된다. 첫 번째 방법은 Kinect 카메라를 사용한다. 두 번째 방법은 Kinect 카메라와 보정 알고리즘을 사용한다. 세 번째 방법은 딥러닝 모델을 사용한다. 네 번째 방법은 딥러닝 모델과 Kinect를 사용한다. 제안 방법을 오차와 처리 속도를 실험한 결과, 네 번째 딥러닝 모델과 Kinect를 동시에 사용하는 방법이 다른 방법에 비해 가장 좋은 결과를 보였다.

Abstract

As the distribution of 3D content such as augmented reality and virtual reality increases, the importance of real-time computer animation technology is increasing. However, the computer animation process consists mostly of manual or marker-attaching motion capture, which requires a very long time for experienced professionals to obtain realistic images. To solve these problems, animation production systems and algorithms based on deep learning model and sensors have recently emerged. Thus, in this paper, we study four methods of implementing natural human movement in deep learning model and kinect camera-based animation production systems. Each method is chosen considering its environmental characteristics and accuracy. The first method uses a Kinect camera. The second method uses a Kinect camera and a calibration algorithm. The third method uses deep learning model. The fourth method uses deep learning model and kinect. Experiments with the proposed method showed that the fourth method of deep learning model and using the Kinect simultaneously showed the best results compared to other methods.

Keyword : Kinect, deep learning, 3D animation, skeleton, human pose estimation

Ⅰ. 서 론

오늘날 AR(Augmented Reality), VR(Virtual Reality)과 같은 3차원 콘텐츠 보급이 증가함에 따라 실시간 컴퓨터 애니메이션 기술의 중요성이 높아지고 있다. 하지만 컴퓨터 애니메이션 제작 과정은 대부분 수작업으로 이루어져 있다. 사실적인 영상을 얻기 위해서는 숙련된 전문가에게도 매우 오랜 시간이 필요하다.

이러한 문제점을 해결하기 위해서 OpenPose[1], DensePose[2], UniPose[3], V2V-PoseNet[4], Integral Human Pose[5], Microsoft(Kinect)[6], ASUS Xtion PRO LIVE[7], ASUS Xtion2[8], LEAP Motion[9] 등과 같은 3차원 정보를 획득 할 수 있는 센서와 딥러닝(deep learning) 기술을 결합한 시스템을 이용한다. 이 시스템은 간단하게 사용자의 동작을 인식하고 애니메이션을 제작하여 비용을 절감한다. OpenMMD[10], VNect[11] Kinect Animation Studio[12], PETE D(Tutorial)[13], Brekel Pro Body2[14]는 센서와 딥러닝을 이용하여 사용자의 동작을 인식하고 애니메이션을 제작하는 시스템이다.

OpenMMD는 카메라 영상과 비디오 영상에서 OpenPose[1], Martinez et al.[15], Laina et al.[16]를 이용하여 사용자의 동작을 인식한다. 그리고 3차원 자체 확장자 애니메이션 파일로 저장한다. 일반 확장자가 아닌 자신들 만의 확장자로 변환되기 때문에 OpenMMD에서 제공하는 시스템에서만 확인 할 수 있다. 또한 실시간 애니메이션 변환이 불가능하다. VNect은 딥러닝 모델을 이용하여 1개의 RGB카메라에서 사용자의 동작을 인식한다. 그리고 애니메이션을 실시간 제어하는 시스템이다. 하지만 애니메이션을 저장하고 추출하는 기능은 아직 없으며 사전 준비가 필요하다. Kinect Animation Studio는 간단하게 Kinect 관절 데이터를 Autodesk FBX 파일로 내보내는데 사용할 수 있는 시스템이다. 하지만 FBX파일을 확인하기 위해서는 Maya, Max, CINEMA 4D 등과 같은 새로운 시스템을 사용한다. 그리고 바닥 평면에 정확하게 정렬되지 않아 거리 이동에 따라 애니메이션의 위치가 변동된다. PETE D는 Kinect의 Orientations 데이터를 Direxr X를 사용해 실시간으로 시각화하여 애니메이션을 보여주는 시스템이다. 하지만 Kinect의 데이터를 저장할 수 없고 실시간으로 동작하는 애니메이션만 보여준다. Brekel Pro Body2는 Kinect SDK를 사용하여 사용자 동작을 인식한다. 그리고 FBX, BVH, CSV, BPC , TXT와 같은 확장자로 파일을 저장 할 수 있다. 바닥과 자동 정렬되며 검토할 수 있는 미리보기 옵션도 제공한다. 하지만 139$ ~ 150$를 지불하여 사용한다. 또한 빠르고 심하게 움직이거나 뒤를 도는 등의 행동에 있어서는 정확도가 낮다.

본 논문에서는 딥러닝 모델과 Kinect 카메라 기반 FBX 형식의 애니메이션 제작 시스템 FAVE(FBX Auto Viewer Engine)를 이용하여 자연스러운 인체 움직임을 구현하는 4가지 방법에 대해 연구한다. 첫 번째 방법은 Kinect 카메라를 이용하는 방법으로 Kinect에서 제공하는 관절 정보를 이용하여 애니메이션을 제작한다. 두 번째 방법은 Kinect에서 제공하는 관절 정보를 BSL(Body Segment Length), ROM (Range of Motion) 알고리즘을 통해 보정하고 애니메이션을 제작한다. 세 번째 방법은 RGB 이미지에서 딥러닝 모델을 이용해 관절 정보, 관절 회전 값, 깊이 정보를 추정하고 애니메이션을 제작한다. 네 번째 방법은 Kinect에서 제공하는 깊이 정보와 딥러닝 모델을 이용해 관절 정보, 관절 회전 값을 추정하고 애니메이션을 제작한다. 제안 방법 오차를 실험한 결과 네 번째 방법인 딥러닝 모델과 Kinect를 동시에 사용하는 방법이 다른 3가지 방법에 비해 가장 좋은 결과를 보여줬다.

a) 서울과학기술대학교 정보통신미디어공학전공(Dept. of Information Technology and Media Engineering, The graduate School of Nano IT Design Fusion, Seoul National University of Science and Technology)
b) 서울과학기술대학교 미디어IT공학과((Dept. of Media IT Engineering, The Graduate School, Seoul National University of Science and Technology)
‡ Corresponding Author : 박구만(Goo-man 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 연구

1. Kinect

Kinect는 RGB카메라, IR카메라로 구성된 복합 장치로서

깊이 이미지 및 3차원 좌표를 인식할 수 있는 카메라이다. Kinect는 세 가지 버전을 출시했으며 외관 뿐 아니라 버전에 따라 여러 차이점을 가지고 있다. 여기서 가장 중요한 차이점은 깊이를 측정하는 방법이다. Kinect v1은 적외선 패턴을 읽고 패턴의 왜곡에서 깊이 정보를 얻는 Light coding이라는 방식을 사용한다. Kinect v2 , Azure 는 투광한 적외선이 반사되어 돌아오는 시간에서 깊이 정보를 얻는 TOF(Time of Flight)라는 방식을 사용한다. Kinect는 삼각측량 방법을 사용하여 공간상의 좌표를 취득하며 2차원 픽셀좌표에서 3차원 공간좌표로 재구성할 수 있다[17]. Kinect는 3가지 좌표 공간을 가지고 있다.

첫 번째는 World Space로 마이크로소프트 Kinect 공식 문서에서는 Camera Space라고 정의 했다. 이 공간은 실제의 3차원 World Space로 원점(x=0, y=0, z=0)은 [그림 1]과 같이 Kinect의 IR센서 중앙을 중심으로 한다. x좌표는 센서의 왼쪽(+), 오른쪽(-)으로 측정한다. y좌표는 센서의 위쪽(+), 아래쪽(-)을 측정한다. z좌표는 센서 방향(+)을 측정한다. World Space는 미터 단위를 가지고 있다[19].

그림 1. Kinect World Space 좌표 시스템[18]
Fig. 1. Kinect World Space Coordinate System[18]

두 번째는 Depth Space로 각 픽셀에는 깊이 정보가 들어있다. 원점은 왼쪽-위 코너로 x축은 오른쪽으로 향할수록 커지며 y축은 아래로 향할수록 커진다. 각 픽셀의 깊이 값은 밀리미터 단위이며 유효한 깊이 범위는 500mm ~ 45000mm이다.

세 번째는 Color Space로 각 필셀에는 컬러 정보가 들어있다. 원점은 Depth Space와 마찬가지로 왼쪽-위 코너로 x축은 오른쪽으로 향할수록 커지며 y축은 아래로 향할수록 커진다. API에서 직접 검색한 값은 RGBX형식으로 저장된다. x채널은 항상 예약되어 있으며 0xff이다. 데이터 어레이는 CV_8UC4 형식의 OpenCV cv::Mat에 직접 매핑할 수 있도록 저장된다.

2. FBX

FBX SDK(Software Development Kit)는 Autodesk에서 만든 3차원 데이터 형식 및 이를 다루는 도구 집합으로 3차원 데이터를 크게 Scene, Object, Node, Property, connection, Attribute, Layer의 개념들로 추상화한다. FBX SDK를 이용하여 Scene에 배치한 오브젝트들은 모두 Node로 표현된다. Node의 구체적인 자료들은 Attribute에 들어가 있다. 기하학 정보를 기술하기 위한 Layer는 따로 존재한다. Object들의 설정 값을 읽고 쓰기 위해서는 Property를 이용해야 한다. 관계를 규정하기 위해서는 connection을 사용해야한다.

FBXScene 클래스는 mesh, skeleton, camera, light, animation 등과 같은 다양한 요소들을 FbxNodeAttribute의 하위 클래스로 추상화한다. 여기서 [그림 2]에 나타난 mesh는 polygon으로 만들어진 기하 정보이다. [그림 2]에 나타난 skeleton은 인체의 골격을 유추한 것으로 각각의 관절은 관절들의 부모 자식 관계를 가진다. 이런 3차원 데이터를 제어하기 위해서는 우선 오브젝트가 위치할 FbxManager을 선언한 뒤에 FbxScene를 생성하여 오브젝트를 할당한다. 그 다음 Fbx Node를 선언해 Skeleton들을 생성하고 관절들의 부모 자식 관계를 지정하여 골격 형태를 만들고 mesh를 입혀 모델을 완성한다. 이때 각 관절의 속성과 위치, 크기, 회전 정보 등을 설정한다. 마지막으로 Time 및 key Index를 지정해 모델의 애니메이션을 구현할 수 있다.

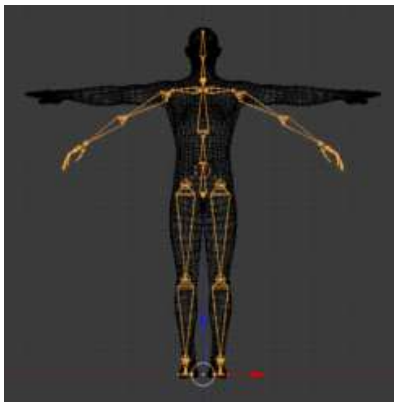


그림 2. FBX 형식의 mesh 그리고 Skeleton[19]
Fig. 2. mesh and Skeleton of FBX format[19]

3. 딥러닝을 이용한 관절 추정

딥러닝을 이용하여 사람의 관절를 추정하는 방식은 Top-down방식과 bottom-up방식으로 나눌 수 있다. Top-down 방식은 영상에서 사람을 검출하고 관절를 추정하는 방식이다. 이 방식은 정확도가 Bottom-up 방식에 비해 높지만 사람을 검출하지 못할 경우 관절 추정 자체를 하지 못하며 다수의 사람이 영상에 있을 경우 느리다는 단점이 있다. bottom-up방식은 영상에 포함된 사람의 관절 포인트를 추정하고 관절 포인트들 간의 상관관계를 이용하여 관절를 추정하는 방식이다. 정확도는 top-down 방식에 비해 떨어지지만 관절 추정 속도가 빠르다.

3.1 관절 추정을 위한 데이터셋

딥러닝 모델을 학습하기 위해서는 데이터셋이 필요하다. 데이터셋의 특성에 따라 모델의 성능이 달라지기 때문에 효율적으로 데이터셋을 선택해야 한다. 사람의 관절 추정을 위해서 사용되는 데이터셋은 LSP(Leeds Sports Pose)[20], MPII Human Pose[21], MS COCO, BODY_25[22], AI Challenger[23], Human3.6M[24], CMU Panoptic[25] 등이 있다.

LSD 데이터셋은 Flickr(사진 공유 사이트)에서 단일 인물이며 스포츠 경기 중인 이미지를 수집하여 만든 데이터셋으로 2,000장의 사진을 가지고 있다. 이미지는 인물의 길이가 약 150px가 되도록 크기가 조정되었다. 각 이미지에는 14개의 관절 좌표가 있다. MPII Human Pose 데이터셋은 약 4만 명의 인물이 포함된 2만 5천장의 이미지로 구성된 데이터 셋이다. 이미지는 유튜브 비디오에서 추출되었다. 각 이미지에는 16개의 관절 좌표와 함께 410개의 활동 레이블링이 제공된다. MS COCO는 15만 명의 인물이 포함된 6만장의 이미지로 구성된 데이터 셋이다. 각 이미지에는 17개의 관절 좌표가 있다. 25개의 관절로 확장한 BODY_25도 있다. AI Challenger 데이터 셋은 약 70만 명의 인물이 포함된 30만장의 이미지로 구성된 데이터 셋이다.

본 논문에서는 OpenPose의 가장 적합한 데이터 셋을 정하기 위해 각각 다른 데이터 셋을 이용하여 학습하여 3차원 애니메이션을 만들어 오차를 비교했다. 가장 좋은 성능은 [그림 3]에서 보듯 AI Challenger 데이터 셋이 가장 좋은 성능 결과를 보였다. 하지만 AI Challenget의 경우 14개의 관절만을 지원하기 때문에 두 번째로 성능이 좋은 BODY_

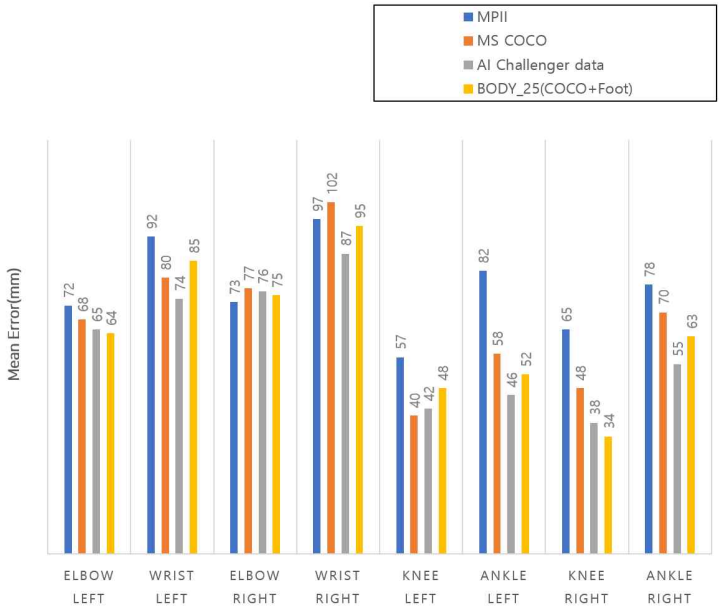


그림 3. 데이터셋에 따른 OpenPose 성능 평가
Fig. 3. Evaluate OpenPose performance based on datasets

25를 사용하여 OpenPose를 학습하고 사용하였다.

3.2 딥러닝을 이용한 2차원 관절 추정

사용자의 2차원 관절을 추정하는 연구는 DeepPose[26] Convolutional Pose Machines[27] OpenPose[1] CrowdPose[28] 등이 있다. OpenPose는 여러 사람의 몸통, 발, 손, 얼굴의 키 포인트를 추정하는 네트워크로 기존 Bottom-up방식의 다중 관절 추정 네트워크보다 높은 성능을 보여준다. OpenPose는 RGB 이미지를 입력으로 받으면 각 관절의 위치를 나타내는 confidence map을 찾고 Part Affinity Fields를 이용하여 각 관절간의 연관성을 찾는다. 연관성과 정확도를 더 높이기 위해 관절의 위치가 올바른지를 인코딩하는 필드를 반복적으로 수행한다. Part Affinity Fields는 각 관절의 위치 정보와 방위 정보를 가지고 관절의 주인을 추정하는 방법으로 기존의 방법보다 더 높은 정확도를 보여준다.

3.3 딥러닝을 이용한 3차원 관절 추정

사용자의 3차원 관절을 추정하는 연구는 Martinez et al.[15] Chen et al.[29] V2V-PoseNet[4] 등이 있다. Martinez et al. 제안한 논문은 이미지를 입력하여 3차원 관절 좌표를 추정하는 것이 아닌 2차원 관절 좌표 값을 입력하여 3차원 관절 좌표를 추정하고 residual connection 추가 및 배치 정규화를 사용하는 방법을 제안한다. 2차원 관절 추정 알고리즘 결과를 바탕으로 3차원으로 변환하는 것을 학습시킴으로써 성능을 더욱 향상시키고 3차원 관절 추정 시스템의 오류 분석을 가능하게 했다.

Ⅲ. 실시간 관절 애니메이션 제작 및 표출 시스템

1. 시스템 개요

본 논문에서 제안하는 FAVE는 딥러닝 모델과 단일 Kinect를 이용하여 사람의 관절을 측정하고 보정해 애니메이션을 제작, 표출하는 시스템이다. 이때 각 환경적 특성과 정확도를 고려하여 애니메이션을 제작 할 수 있도록 4가지 동작 방법을 제안한다. 동작하는 4가지 방법은 하나의 시스

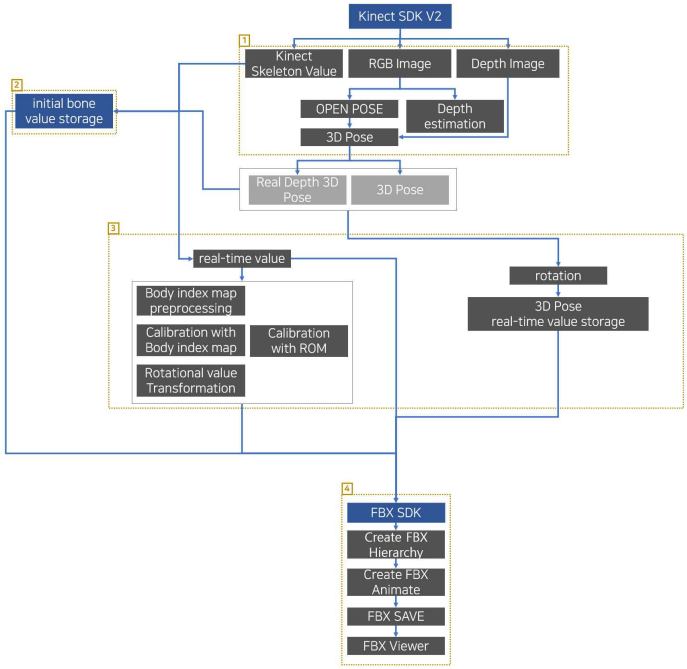


그림 4. FAVE의 설계구조
Fig. 4. FAVE Design Structure

템에서 동작하며 각 방법에 따라 모듈의 활성화 여부가 결정된다. 시스템의 흐름은 [그림 4]와 같이 진행되며 총 4개의 동작 영역을 가지고 있다. 첫 번째 영역은 Kinect에서 RGB Image, Depth Image값을 받아 관절을 추정하는 영역이다. 두 번째 영역은 관절의 계층 설정을 위해서 필요한 초기 관절 정보를 저장하고 보정하는 영역이다. 세 번째 영역은 실시간으로 들어오는 관절 정보를 보정하는 영역이다. 네 번째 영역은 보정된 관절 정보를 이용하여 애니메이션을 생성 저장하는 영역이다.

2. 동작 방법 공통 모듈 설명

2.1 3차원 관절 생성과 계층 설정 모듈

FBX SDK를 이용해 노드와 애니메이션을 제작하기 위해서는 계층구조로 이루어진 관절 계층을 생성해야한다. 계층 설정을 위해서는 3차원 관절이 필요하다. Kinect를 사용하는 경우 추정된 관절의 World Space 값을 초기 관절로 사용한다. 딥러닝 모델을 사용하는 경우 RGB Image에서 2차원 관절을 추정하고 3차원 관절을 추정하여 초기 관절로 사용한다. 관절 계층은 아래 [그림 5]와 같은 형태로 만들어진다. 각 관절은 부모-자식 관계를 가지고 있다. 화살표가 시작하는 부분이 부모 노드이고 화살표가 끝나는 부분이 자식 노드이다. 관절 계층에는 각 관절의 Transform 값이 필요하다. 이 값은 초기 관절 정보 값을 사용한다.

2.2 관절 정보 저장

보정과 생성이 끝난 관절 정보는 일정한 프레임으로 저장이 되며 저장되는 정보는 각 관절의 관절 번호, 측정된 위치 좌표(X, Y, Z), 회전 좌표(X, Y, Z, W), 실행 프레임 정보 순으로 저장된다. 일정한 프레임으로 정보를 저장하는 이유는 FBX 변환 시 애니메이션의 시간 간격이 달라 심한 흔들림 현상을 가져올 수 있기 때문이다. 또한 저장되는 정보는 키넥트 카메라 각도에 따른 회전 보정을 거친다.

2.3 애니메이션 생성

FBX SDK는 처음 관절 정보가 들어오면 초기에 설정한 계층 구조를 기반으로 관절의 유형, 관절의 초기 방향, 위치, 크기를 설정하여 FBX Node를 만든다. 그리고 들어온

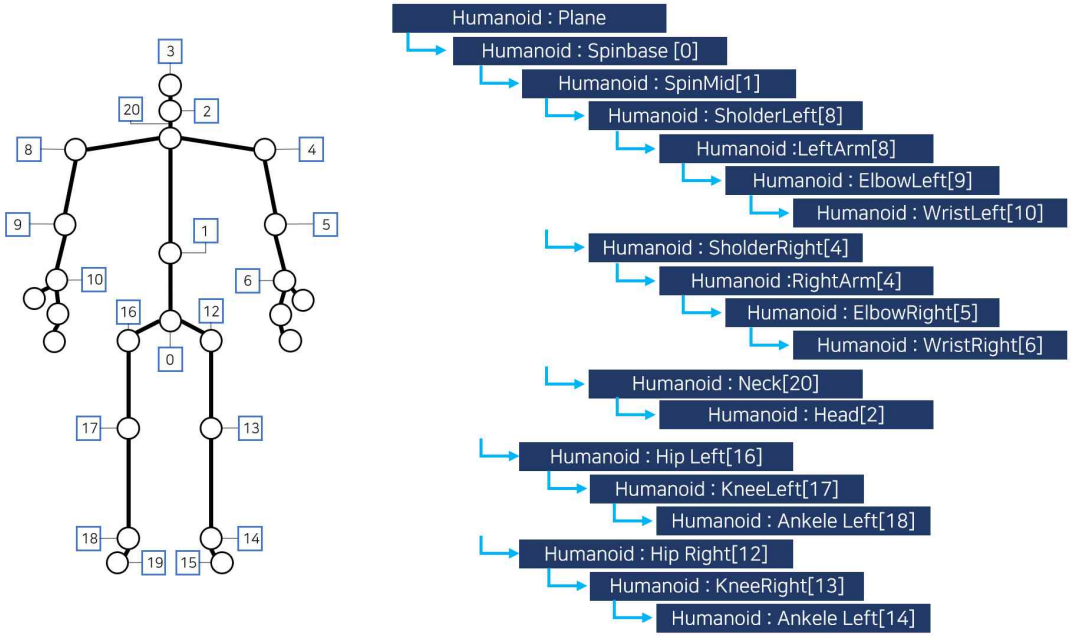


그림 5. FBX 생성을 위한 관절 계층 구조
Fig. 5. Joint hierarchical structure for FBX generation

관절 정보를 이용해 Node에 저장한다. 순서는 다음과 같다. 첫 번째로 관절 정보들의 실행 시간을 비교하여 Node에 저장할 Time을 계산한다. 두 번째로 관절의 특성에 따라 관절의 회전 값을 회전행렬로 변경하고 애니메이션에 사용될 수 있도록 FBXAnimCurve에 값을 넘겨준다. 넘겨받은 Time와 Curve를 기반으로 애니메이션을 제작한다. 여기서 관절의 특성은 다음과 같다. [그림 5]에서 관절 4, 5, 9, 8, 12, 13, 16, 17은 자식 관절이 각 1개씩 존재하는 관절로 자식 회전 값을 사용한다. 관절 3, 6, 10, 14, 18은 자식 관절이 없는 관절로 관절의 회전 값을 전달하지 않아도 된다. 관절 20은 자식 관절이 1개 이상을 가지고 있는 관절로 초기 회전 값을 사용한다. 관절 0은 자식 관절이 1개 이상을 가지고 있는 관절이지만 자식 관절에게만 영향을 미치는 것이 아닌 모든 관절에 미치는 관절로 회전 값이 아닌 위치 값을 사용하는 관절이다.

3. 애니메이션 제작 방법

3.1 Kinect와 관절 보정을 사용하는 방법

Kinect에서 제공하는 정보는 노이즈 값이 존재한다. 또한 폐색과 인물 주변의 환경에 영향을 많이 받는다. 이에 Kinect에서 제공하는 관절 정보를 보정해 애니메이션을 제작하는 방법이다. 관절 중 위치추정 오류가 가장 높은 8개 [그림 5]에서 5, 6, 9, 10, 13, 14, 17, 18의 관절들을 BSL, ROM 알고리즘을 이용해 보정한다[29]. 보정은 [그림 6]과 같은 순서로 진행된다. BSL알고리즘을 적용하기 위해서는 인체 깊이 지도를 생성해야한다. 깊이 지도는 3차원 장면을 0부터 255까지의 값을 지닌 화소들로 변환하여 나타낸 것이다. 인체 깊이 지도는 잡음이 많아 닫힘(close)연산과 중간 값(median blur)필터를 이용해 전처리한다. 인체 깊이 지도가 생성되면 관절의 2차원 위치 좌표의 깊이지도 화소값에 접근하여 윤곽이 벗어났는지 추정한다. 그리고 관절이 보정될 유력 후보 위치들이 모인 관심 영역을 지정한다. 관심영역의 위치 좌표 중에서 자신에 대응되는 3차원 좌표가 BSL에 만족하는 점들을 찾아낸다. 3차원 관절 계층 생성에서 얻어진 정보 중 보정 대상 관절에 해당되는 정보와 자식 관절 위치, 부모 관절위치 거리를 이용하여 최적의 위치를 보정 대상 관절의 최종적인 보정 위치로 설정하여 보라색 점으로 나타낸다.

Kinect가 제공하는 인체 관절의 회전 값은 보정 전 잘못된 위치 좌표에 대한 정보이므로 보정된 위치 좌표에 대한 새로운 회전 값을 추정해야 한다. 첫 번째로 보정된 관절 위치 $P_j^{'}$와 이것을 연이은 자식관절의 위치 P_{j+1}, P_{j+2} 총 세 점을 카메라 좌표계에서 부모 관절을 기준으로 한 좌표계로 변환한다. 두 번째로 $\overrightarrow{P_j^{'}P_{j+1}}$와 $\overrightarrow{P_{j+1}P_{j+2}}$ 사이의 각

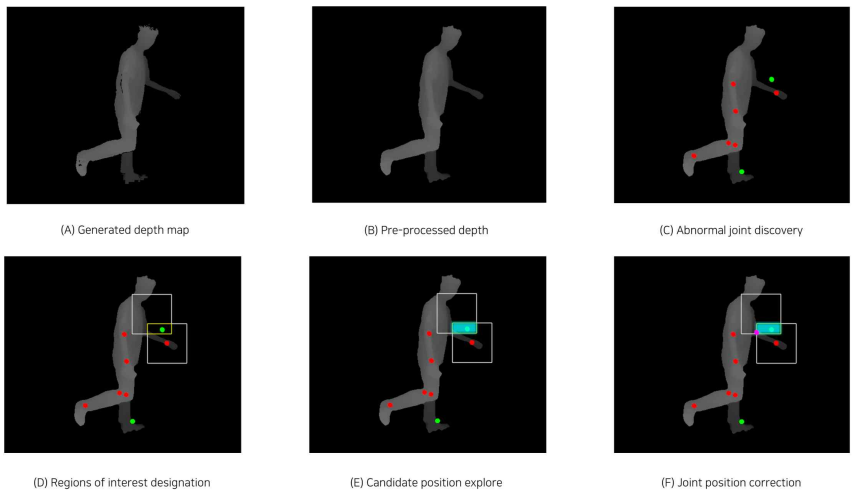


그림 6. BSL을 이용한 관절 보정 절차
Fig. 6. Joint correction procedure using BS

θ을 구한 뒤, 원본 형태(회색 점) 내의 직선 $\overline{P_j^{'}P_{j+1}}$의 위치를 [그림 7]과 같이 표준 골격 형태(흰색 점)에 정렬된(체크무늬 점) 직선 $\overline{P_j^{'}P_{j+1}^{'}}$로 변환하고 $\overline{P_{j+1}P_{j+2}}$의 위치도 $\overline{P_j^{'}P_{j+1}^{'}}$와 θ을 유지하도록 하는 $\overline{P_{j+1}^{'}P_{j+2}^{'}}$로 변환한다.

마지막으로 표준 골격 형태 $\overline{P_j^{'}P_{j+1}^{'}}$와 원본 형태 $\overline{P_j^{'}P_{j+1}}$ 사이의 회전각 $R_1(X_1, Y_1, Z_1)$을 구한다. 그리고 $P_{j+2}^{'}$을 R_1 만큼 회전시킨 점 $P_{j+2}^{''}$을 구한다. $\overrightarrow{P_{j+1}P_j^{'}}$와 $\overrightarrow{P_{j+1}P_{j+2}^{''}}$간의 법선 벡터 $\overrightarrow{n_1}$, 그리고 원본 형태에서의 벡터 $\overrightarrow{P_{j+1}P_j^{'}}$와 $\overrightarrow{P_{j+1}P_{j+2}}$간의 법선 벡터 $\overrightarrow{n_2}$을 구하고 두 법선 벡터 사이의 각 $R_2(X_2, Y_2, Z_2)$을 구한다. 관절의 표준 골격 형태에서의 각 R_0과 R_1, R_2로부터 최종 회전 값 $R_j^{'}$을 구한다.

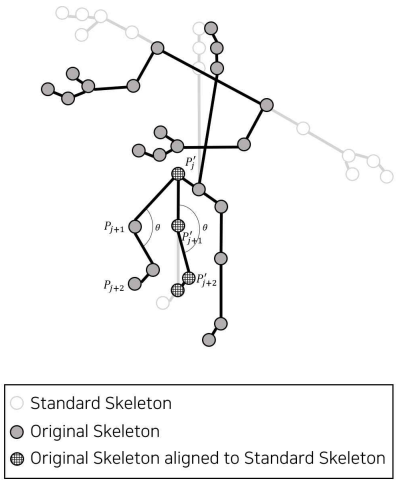


그림 7. 표준 골격 형태로의 관절 위치 변환
Fig. 7. Transforming the position of the joint into a default skeleton

3.2 딥러닝을 사용하는 방법

Kinect의 관절 추정은 깊이 지도에서 랜덤 포레스트 알고리즘을 이용하여 추정된다. 때문에 [그림 8]과 같이 관절 겹침 현상으로 인한 깊이지도 폐색이 일어난 경우에 추적이 힘들다. 또한 앞, 뒤, 좌, 우의 특징을 구분하지 못하므로 정교한 관절 추적이 힘들다. 3.1에서 설명한 보정 방법을 사용한다 하더라도 특징점을 찾아 보정하는 것이 아니기 때문에 보정에 실패할 가능성이 존재하다. 이에 딥러닝을 이용하여 사람의 관절 특징을 찾아 관절을 추정하고 3차원으로 변환하여 애니메이션을 제작하는 방법이다.

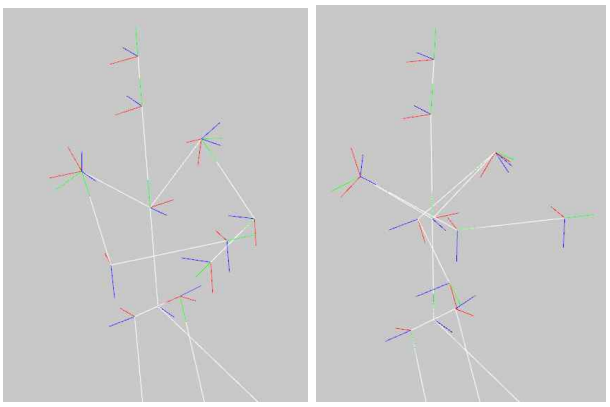


그림 8. 폐색 데이터를 이용한 애니메이션 생성 (좌) - 딥러닝을 이용한 방법 (우) - 키넥트를 이용한 방법
Fig. 8. Animation generation with occlusion data (left) - Deep learning method (right) - kinect method

3차원 관절 추정, 3차원 깊이 정보, 관절 회전 값 추정은 하나의 네트워크로 추정하는 것이 불가능하기 때문에 [그림 9]와 같은 순서로 진행된다. 2차원 관절 추정은 Open-Pose[1]를 이용한다. 3차원 관절 추정은 2차원 관절 추정 결과를 Martinez et al[15] 제안한 네트워크에 입력 값을 넣어 추정한다. 이 네트워크는 Linear 연산과 batch norm, ReLU, dropout을 하나의 선형 레이어로 간주하여 두 개의 선형 레이어 결과와 입력 값을 더함으로써 하나의 residual 블록을 형성했다. 이 residual 블록을 두 번 반복하고, 그림엔 표시되지 않았지만 입력 다음과 출력 직전에 선형 레이어를 한 번씩 더해 총 6개의 레이어를 만들어 2차원 관절에서 3차원 관절을 추정한다.

3차원 관절 추정 z값은 실제 3차원 좌표의 값이 아닌 x, y 좌표를 이용하여 추정된 상대적인 z값이다. 때문에 애니메이션을 제작하기 위해서는 실제 z값 추정해야 한다. 실제 z값을 추정하기 위해서는 Laina et al.[16]가 제안한 네트워크를 사용한다. 이 네트워크는 인코더-디코더로 구성되어 RGB 이미지에서 깊이 정보를 추정 한다. 최적의 예측을 위해 전역 및 로컬 모두 capture한다. ResNet50은 네트워크

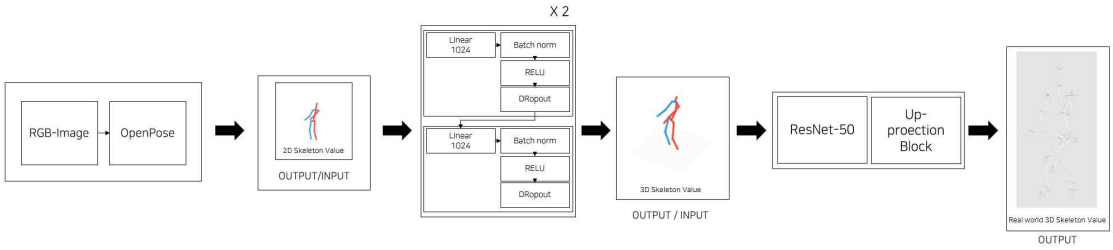


그림 9. 딥러닝을 이용한 방법의 관절 추정 흐름도
Fig. 9. Joint Estimation Flowchart of Methods Using Deep Learning

의 인코더 부분에 사용된다. up projection은 디코더 부분에 사용된다.

3.3 딥러닝과 Kinect를 사용하는 방법

딥러닝을 이용하여 2차원 관절, 3차원 관절, 깊이 정보를 추정할 경우 많은 연산 과정이 필요하다. 때문에 오랜 시간이 걸려 실시간으로 애니메이션을 제작할 수 없다. 또한 딥러닝을 이용하여 추정한 깊이 정보의 경우 노이즈가 생긴다. 노이즈가 생긴 깊이 정보는 관절 회전 값과 애니메이션 결과 값에 영향을 미치게 된다. 이에 깊이 정보를 키넥트로 대체하여 관절을 계산함으로 연산 과정을 줄이며 관절의 정확도를 높인다. 관절 추정은 [그림 10]과 같이 진행된다. 3.2에서 제안한 방법과 마찬가지로 2차원 관절 추정 후 결과를 바탕으로 3차원 관절을 추정한다. 그리고 3차원 관절의 z값을 딥러닝을 통해서 추정하는 것이 아닌 Kinect의 깊이 정보를 불러와 대체한다.

Ⅳ. 실 험

본 연구에서는 적외선 센서, 깊이 센서, 컬러 카메라를 지원하는 Kinect SDK 2.0을 이용했으며 BSL 및 알고리즘 개발은 C/C++를, 그리고 딥러닝 모델 학습 및 추정은 python, pytorch를 사용했다. 또한 딥러닝 모델은 CUDA 10.1에서 실행되며 실시간 3차원 애니메이션 제작 및 표출은 FBX SDK 2019, OpenFrameWork 0.9.8로 자체 개발된 프로그램을 이용하였다. 그래픽 카드는 NVIDA Geforce RTX 2080 with Max-Q를 사용했다.

본 실험은 인체의 관절에 마커를 부착한 뒤 전체의 시스템을 실행한다. 그리고 Kinect 카메라 앞에서 특정한 동작을 수행한 후 저장된 1924개의 프레임에 4가지 방법들을 적용하여 애니메이션을 제작하고 관절의 오차를 비교한다. 관절의 오차는 오류가 잦은 8개의 관절을 대상으로 한다. 오차는 마커를 이용해 제작된 애니메이션과 4가지 방법들로 추정하여, 제작된 애니메이션과의 길이 차이를 비교하여 계산한다. 애니메이션 제작 결과는 [표 1]과 같다. 애니메이션 제작 결과 폐색이 없고 단순한 동작들은 모든 방법들이 정상적으로 제작되는 것을 볼 수 있었다. 또한 Kinect 깊이 이미지와 딥러닝에서 추정된 깊이 이미지의 차이로 관절의 위치 값과 회전 값이 다르게 적용된 것을 볼 수 있다. 폐색이 있는 경우 Kinect를 사용한 방법은 관절이 뒤틀리는 것을 볼 수 있다. 그리고 BSL, ROM을 이용해 보정하

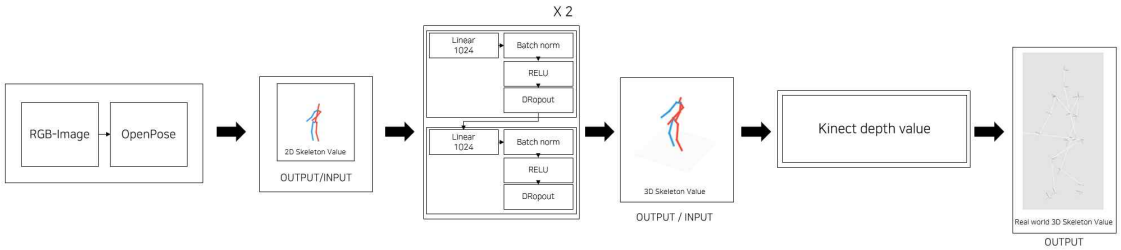
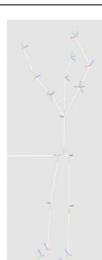
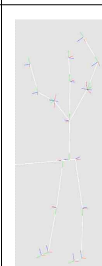


그림 10. 딥러닝과 Kinect를 이용한 방법의 관절 추정 흐름도
Fig. 10. Joint Estimation Flowchart of Methods Using Deep Learning and Kinect

표 1. 수행 동작과 실험결과
Table 1. User Action and Experimental Results

Input Image	Using Kinect	Using the Kinect and BSL, ROM	Using Deep Learning Model	Using Deep Learning Model and Kinect
				
				
				
				
				

는 경우에도 완벽하게 입력 이미지와 동일하게 애니메이션을 제작하지 못하는 것을 볼 수 있다.

1. 오차

실험 결과 각 관절의 오차는 [그림 11]과 같다. Ⅰ~ Ⅷ는 순서대로 [그림 5]의 관절 번호 5, 6, 9, 10, 13, 14, 17, 18번이다. 본 실험은 단일 Kinect로 진행되어 좌측, 우측, 회전의 경우 관절의 오차가 상당히 커지는 것을 볼 수 있었다. 또한 관절의 오차에서 두 가지 특징을 볼 수 있었다. 첫 번째 특징은 다리 관절보다 팔 관절에서 오류가 높게 나타난 것이다. 다리 보다 팔에서 더 높은 오류가 나타난 것은 격한 움직임을 보이더라도 상대적으로 팔보다 운동할 수 있는 범위가 적으며 폐색에 한계가 있기 때문이다. 두 번째 특징은 부모의 관절 보다 자식 관절의 오류가 높게 나타난 점이다. 부모의 관절 보다 자식 관절의 오류 값이 높게 나타난 이유는 부모 관절의 움직임에 따라 자식 관절도 함께 움직이기 때문이다.

각 방법에 따른 오차는 [표 2]와 같다. 가장 오차가 큰 방법은 Kinect를 사용한 방법으로 딥러닝과 Kinect를 사용한 방법보다 2배 높은 오차를 보였다. Kinect 관절 값을 사용한 방법보다 딥러닝 모델에 추정한 관절 값을 사용한 방법이 더 낮은 오차를 보였다. 그 이유는 Kinect의 경우 관절의 겹침, 폐색, 주변 환경에 강건하지 못했기 때문이다. 또한 보정 알고리즘을 사용한 경우에도 오류가 오랜 시간 지속될 경우 보정에 한계가 있기 때문이다.

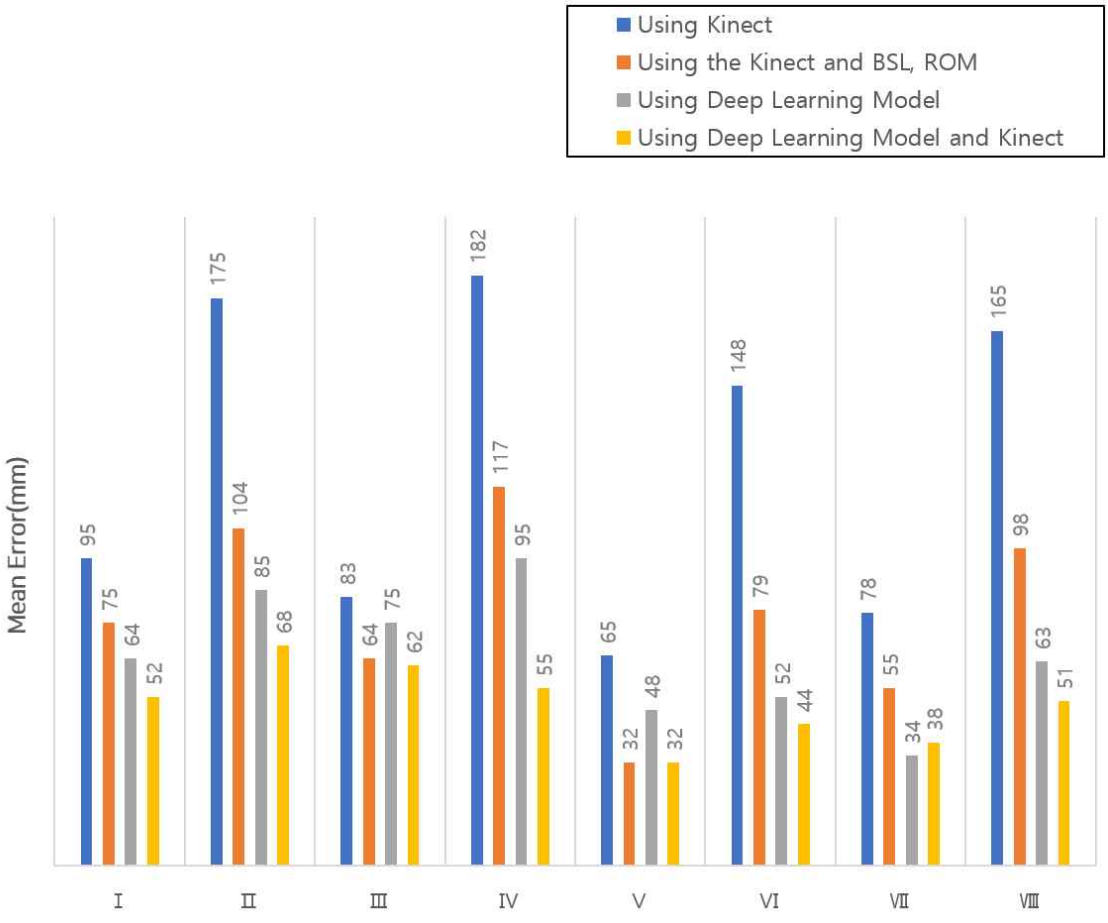


그림 11. 8개의 관절 오차
Fig. 11. 8 joint errors

표 2. 방법에 따른 오류
Table 2. Errors according to methods

Methods	Mean error(mm)
Using Kinect	123
Using the Kinect and BSL, ROM	78
Using Deep Learning Model	64.5
Using Deep Learning Model and Kinect	50.2

2. 호환성

각 방법에 따른 제작 파일 호환성은 [표 3]과 같다. 딥러닝 모델은 실시간으로 애니메이션 제작이 가능하며 일반 이미지, 영상 파일을 다운로드 받아 애니메이션 제작도 가능하다. 또한 딥러닝 모델을 변경할 수 있어 다양한 모델을 사용하여 환경에 맞춰 대응할 수 있다. Kinect는 실시간으로 애니메이션 제작만 가능하다. 각 모드에 따른 필요 장비는 [표 4]와 같다. Kinect를 사용하는 방법은 Kinect와 일반 사양의 컴퓨터가 있다면 사용할 수 있다. 하지만 딥러닝을 사용하는 모드는 고사양의 컴퓨터가 필요하다.

V. 결 론

오늘날 3차원 콘텐츠 보급이 증가함에 따라 실시간 애니메이션 기술의 중요성이 높아지고 있다. 하지만 사실적인 영상을 얻기 위해서는 숙련된 전문가도 오랜 시간이 필요하다. 이러한 문제점을 해결하기 위해서 센서와 딥러닝 기술을 이용하여 사용자의 동작을 간단히 인식하는 시스템이 연구되고 있다. 본 논문에서 딥러닝 모델과 Kinect 카메라 기반 FBX 형식의 애니메이션 제작 시스템 FAVE를 개발하고 자연스러운 인체 움직임을 구현하는 4가지 방법에 대해 연구했다. 첫 번째 방법은 Kinect 카메라를 사용하고 두 번째 방법은 Kinect 카메라와 보정 알고리즘을 사용한다. 세 번째 방법은 딥러닝 모델을 사용하고 네 번째 방법은 딥러닝 모델과 Kinect를 사용한다. 실험 결과 마커를 이용해 제작된 애니메이션과 4가지 방법들로 추정하여 제작된 애니메이션과의 길이 차이를 측정했다. 측정 결과 오차가 가장 큰 방법은 Kinect를 사용한 방법으로 딥러닝과 Kinect를 사용한 방법보다 2배 높은 오차를 보였다. Kinect를 사용한 방법보다 딥러닝을 사용한 방법이 오차가 더 적은 것을 확인할 수 있었다. 또한 관절의 오차에서는 두 가지 특징이 나타나는 것을 볼 수 있었다. 첫 번째는 다리 관절 보다 팔 관절에서 오류가 높게 나타나는 것이고 두 번째는 부모의 관절보다 자식 관절의 오차가 높게 나타는 점이었다. 향후 연구로는 폐색, 겹침 해결을 위해 다중 Kinect를 지원하여 오차가 높아지는 현상을 최소화 하고 단일 사용자가 아닌 다중 사용자도 애니메이션 제작이 가능하도록 보완할 예정이다.

표 3. 제작 파일 호환성
Table 3. Authoring File Compatibility

Methods	Real-time	Storage	Download
Using Kinect	O	O	X
Using the Kinect and BSL, ROM	O	O	X
Using Deep Learning Model	O	O	O
Using Deep Learning Model and Kinect	O	O	O

표 4. 필요 장비
Table 4. Equipment required

Methods	Kinect	RGB Camera	High performance computer
Using Kinect	O	X	X
Using the Kinect and BSL, ROM	O	X	X
Using Deep Learning Model	X	O	O
Using Deep Learning Model and Kinect	O	X	O

참 고 문 헌 (References)

[1] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh, “OpenPose: realtime multi-person 3D poseestimation using Part Affinity Fields”, In arXiv preprintarXiv:1812.08008, 2018.

[2] R. A. Güler, N. Neverova, and I. Kokkinos. Densepose: Dense human pose estimation in the wild. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.

[3] Bruno Artacho, Andreas Savakis, "UniPose: Unified Human Pose Estimation in Single Images and Videos" Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 7035-7044

[4] Moon, Gyeongsik, Ju Yong Chang, and Kyoung Mu Lee. "V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map." Proceedings of the IEEE conference on computer vision and pattern Recognition. 2018.

[5] Sun, X., Xiao, B., Liang, S., Wei, Y.: Integral human pose regression. arXiv preprint arXiv:1711.08229 (2017)

[6] [Internet] Microsoft Kinect v3 https://azure.microsoft.com/ko-kr/services/kinect-dk/

[7] [Internet] ASUS Xtion PRO LIVe https://www.asus.com/kr/3D-Sensor/Xtion_PRO_LIVE/

[8] [Internet] ASUS Xtion2 https://www.asus.com/kr/3D-Sensor/Xtion-2/

[9] [Internet] LEAP Motion https://developer.leapmotion.com/#101

[10] [Internet] “OpenMMD” https://github.com/peterljq/OpenMMD

[11] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin,M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt, “Vnect: Real-time 3d human pose estimation with a singlergb camera”, In ACM Transactions on Graphics, volume 36,2017.

[12] [Internet] “Instructions”http://marcojrfurtado.github.io/KinectAnimationStudio/usage .html

[13] [Internet] “Avateering with kinect v2 – Joint Orientations” https://peted.azurewebsites.net/avateering-with-kinect-v2-joint-orientations/

[14] [Internet] “Brekel Body v2” https://brekel.com/brekel-pro-body-v2/

[15] J. Martinez, R. Hossain, J. Romero, and J. J. Little. A simple yet effective baseline for 3d human pose estimation. In IEEE International Conference on Computer Vision, ICCV, 2017.

[16] 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, pages 239 – 248. IEEE, 2016.

[17] Jun hee Kim, Sae-Woung Yoo and Kyung-Won Min, Microsoft Kinect-based Indoor Building Information Model Acquisition., Computational Structural Engineering Institute of Korea 31(4), 207-214.

[18] Sang-Joon Kim, “Design and Implementation of Authoring Tool for Dynamic Projection Mapping Content,” Degree thesis (Bachelor's degree), Seoul Media Graduate School: New Media Studies Department 2019.2

[19] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages1653 – 1660. IEEE, 2014.

[20] [Internet] Leeds Sports Pose(LSP) https://sam.johnson.io/research/lsp.html

[21] [Internet] MPII Human Pose http://human-pose.mpi-inf.mpg.de/

[22] [Internet] MS COCO https://cocodataset.org/#home

[23] [Internet] AI Challenger http://dataju.cn/Dataju/web/datasetInstanceDetail/440

[24] [Internet] Human3.6M http://vision.imar.ro/human3.6m/description.php

[25] [Internet] CMU Panoptic http://domedb.perception.cs.cmu.edu/

[26] Alexander Toshev, Christian Szegedy "DeepPose: Human Pose Estimation via Deep Neural Networks" arXiv preprint arXiv:1312.4659, 2013

[27] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In CVPR, 2016

[28] Jiefeng Li, Can Wang, Hao Zhu, Yihuan Mao, Hao-Shu Fang, and Cewu Lu. Crowdpose: Efficient crowded scenes pose estimation and a new benchmark. arXiv preprint arXiv:1812.00324, 2018

[29] J. heon jeong, S. joon kim, M. suk Yoon and G. man Park "Body Segment Length and Joint Motion Range Restriction for Joint Errors Correction in FBX Type Motion Capture Animation based on Kinect Camera ", JBE Vol. 25, No. 3, May 2020

저 자 소 개

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

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

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