나만의 캐릭터로 플레이하는 AR 게임

A반 2조

안지수 김희선 맹인호 양진미 이단디

목차

1. 프로젝트 소개

1.1. 프로젝트 개요

• 프로젝트 주제

프로젝트 주제는 '나만의 3D 캐릭터로 즐기는 custom AR 보드게임'이다. 부제로는 '청년 AI·BigData 아카데미 수료를 위한 여정, PosMarble'이다.

• 메타버스 정의

메타버스는 세계를 뜻하는 '유니버스(Universe)'와 가공, 추상을 뜻하는 '메타(Meta)'의 합성어다. 메타버스의 정의는 현재 제대로 확립되지 않았으며 새로운 주제에 도전하는만큼 본 프로젝트에 걸맞는 메타버스의 정의를 새롭게 정의하는 것이 의미있을 것이라 판단해 메타버스를 새롭게 정의해보았다. 본 조에서는 메타버스를 '현실과 가상을 넘나들며 사용자의 아이덴티티를 표현할 수 있는 세계'로 정의했다. 이 세계 안에서 자신의 아이덴티티가 담겨있는 캐릭터를 등장시키고 해당 캐릭터가 사용자의 습관, 버릇 또는 사용자가 원하는 모션을 취할 수 있게 한다.

1.2. 추진 배경 및 현황

1.2.1. 추진 배경

• 메타버스 시대의 도래

메타버스는 약 30년간의 발전을 통해 꾸준히 진화한 기술이다. 메타버스는 컴퓨팅, 네트워크의 성능, 블록체인 가상화폐의 등장을 수용하면서 놀라운 성장을 이끌어내고 있다. 이에 AI 기술이 더해져 메타버스의 주요 종류인 가상세계, 증강현실을 현실감있게 즐길 수 있게 되어가고 있다. 메타버스는 현재에도 다양한 기술을 결합해 범위를 넓혀가고 있으므로 정확하게 정의하기 어려운 개념이다. 특히 실세계의 제약조건에서 자유로울 수 있는 다양성의 보장은 메타버스의 주요 매력이자 성공 요소이다.

• MZ세대와의 소통

메타버스의 주 소비층인 MZ세대(1980년대 초 ~ 2000년대 초 출생한 밀레니얼 세대와 1990년대 중반 ~ 2000년대 초반 출생한 Z세대)는 개성을 살려 자신만의 무언가를 만들고 싶어하는 욕구가 크고 남들과 다른 점에서 존재 가치를 찾는 특징을 가지고 있다. 또한 Z세대로 갈 수록 전체적인 스토리를 중요시하는 소비형태는 '스토리 소비'에서 각 캐릭터 중심이며 상대적으로 짧은 시간을 요구하는 '캐릭터 소비'로 소비형태가 변화되고 있다. 그러므로 AR 보드게임을 나만의 캐릭터와 함께 즐기면서 자신만의 아이덴티티를 드러내는 현 프로젝트의 목적은 메타버스의 주 소비층인 MZ세대의 특징을 반영한 방향이다.

MZ 세대 특징

다양성 인정
- 타인의 취향 존중
- 삶의 방식을 선택 사항으로 인식

여가 중시, 현실성
- 워라밸(일과 삶의 균형) 중시
- 노력 대비 보상, 현재 중시

환경·윤리적 가치 중시
- 가치관·신념 기반 소비·투자
- 선한 영향력

자기 중심적 소비
- 나만의 스타일 추구
- 맞춤형 선호

디지털 네이티브
- 신기술에 친숙
- SNS 통해 가치관·신념 표현

재미 추구
- 즐기는 소비·투자·일
- 커뮤니티로 취향 공유

자료 : '이코노미조선' 정리

1.2.2. 관련 기술 현황

• AR 시장 전망

2019년에 비해 2030년에 메타버스 시장 전망이 30배 이상으로 증가하는 것으로 예측된다. 그 중에서도 본 프로젝트의 주제이자 메타버스의 한 종류로 분류되는 증강현실(AR)의 전망이 더욱 높은 것을 볼 수 있다.

• AR 보드게임

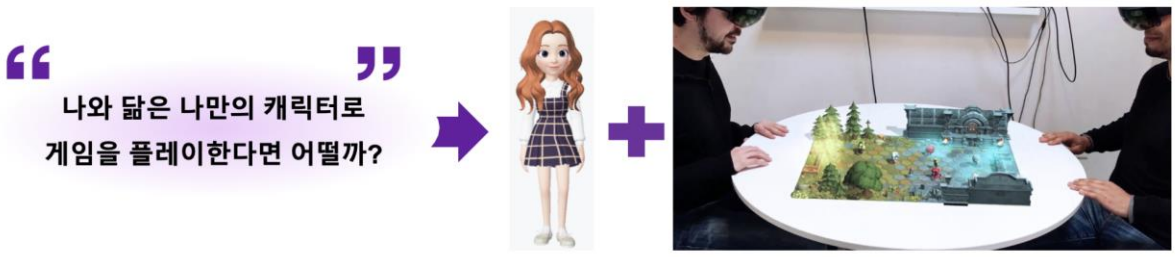
- Monopoly Architect AR

Monopoly Architect AR은 Jo's Works라는 회사에서 개발한 보드게임으로, 3D 프린팅, AR/MR 및 기존 보드 게임을 통합하여 차세대 보드 게임 플레이어를 위한 풍부한 게임 경험을 제공하는 개념 증설 증강현실 보드 게임이다. monopoly게임의 지루함을 AR의 현실감을 커버하겠다는 것이 이 제품의 목적이다. 이 회사 또한 필자가 제안한 기술의 Marker AR 방식을 사용한다.

- Tilt Five

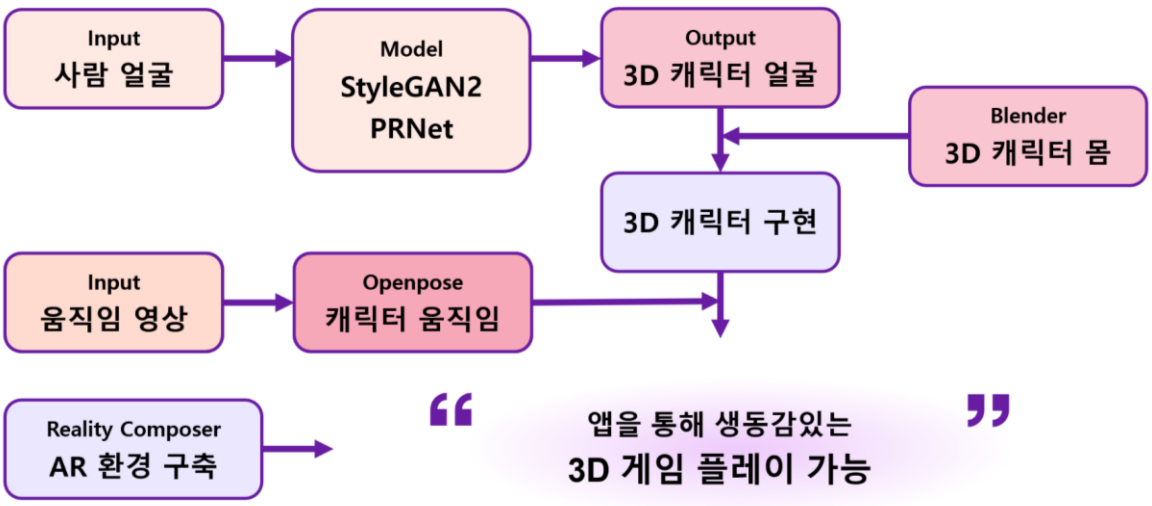
Tilt Five는 보드 게임을 증강현실로 즐기는 제품이다. 구성품은 AR 글래스, 컨트롤러, 게임기의 역할을 하는 스마트폰 앱, USB로 연결하는 보드판이 있다. 앱으로 게임을 실행하고 글래스를 착용하면 빈 보드판이 3D그래픽의 보드 게임으로 변한다.

본 조에서는 현재 시중에서 사용되고 있는 AR 보드게임에 나만의 3D 캐릭터를 추가하여 게임의 주 소비층이 될 MZ세대의 요구사항을 맞추려고 한다.



2. 구현 과정

2.1. 전체 구조도



구조도는 크게 4가지 과정으로 이루어져있다.

첫번째는 사용자의 얼굴을 입력받아 3D 캐릭터 얼굴을 생성하는 과정, 두번째는 딥러닝 모델을 통해 나온 3D 캐릭터 얼굴과 몸을 Blender를 통해 병합하는 과정, 세번째로는 사용자가 원하는 모션을 입력받아 캐릭터에 똑같은 움직임을 입히는 과정, 마지막으로는 Reality Composer를 이용해 AR 환경을 구축하는 과정으로 이루어져있다.

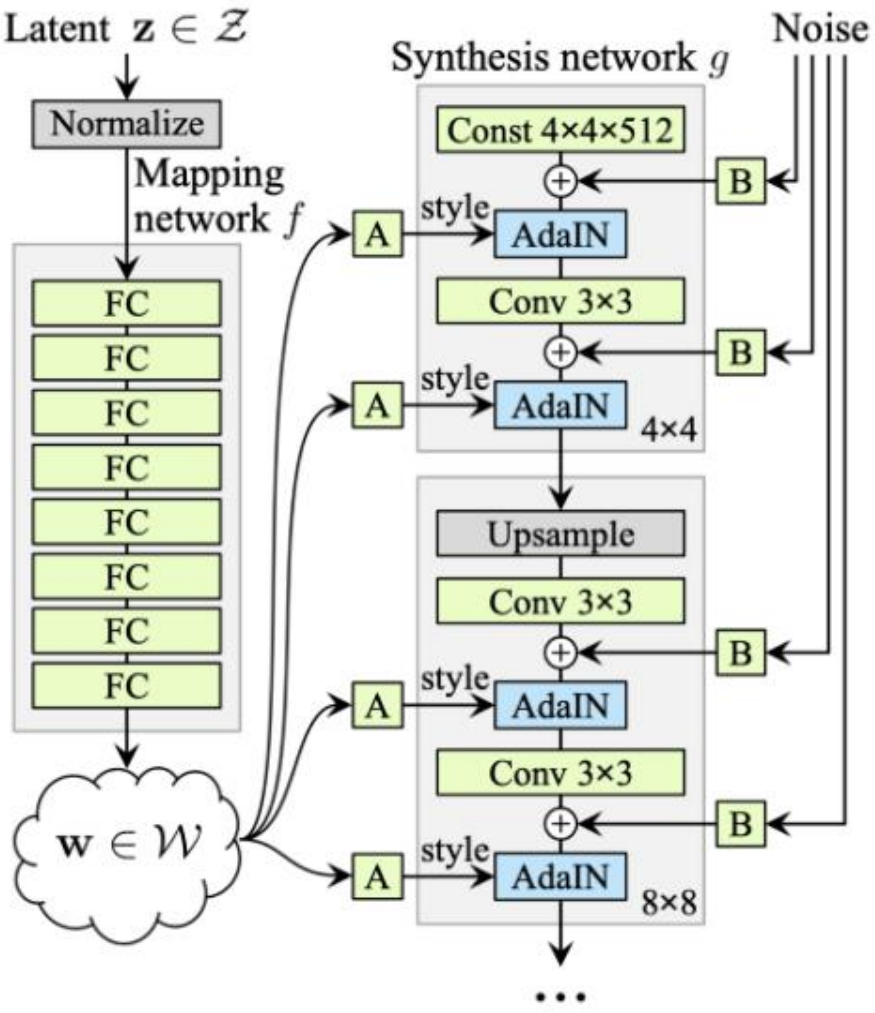
2.2. AI

2.2.1. 2D 캐릭터 얼굴 생성

- StyleGAN2

StyleGAN2(style-based GAN2)를 이용해 원본 얼굴을 캐릭터화된 2D 이미지로 변환해준다.

StyleGAN2는 원하는 화풍의 이미지에 필터를 씌운 것 같은 효과를 주는 딥러닝 모델이다.

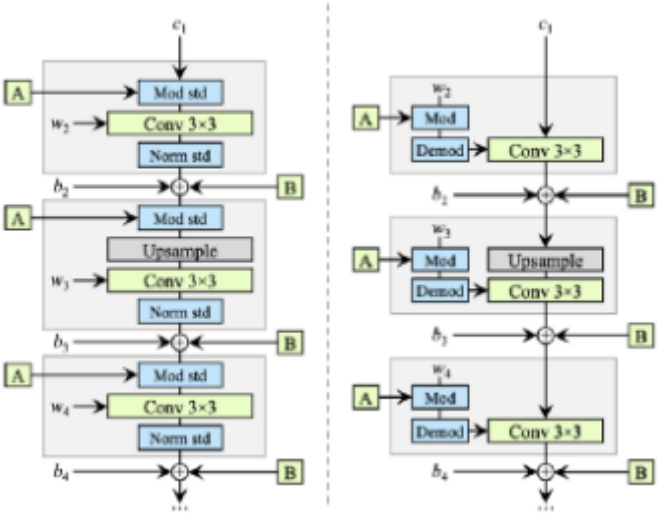


<StyleGAN 생성자 구조도>

StyleGAN의 가장 큰 특징은 다음과 같다.
층을 거듭할수록 해상도를 점진적으로 향상시키는 Progressive Growing 방법을 사용하여 확률적으로 생성된 latent variables가 아닌 고정된 값을 가진 텐서를 생성자의 input으로 넣는다.
확률적으로 생성된 latent variables는 style vector로써 사용되고 이는 비선형의 9개의 FC층을 거친다.

전체 구조도를 보면 4x4x512사이즈의 고정텐서를 인풋으로 받고 층마다 style vectors가 추가된 형태로 model architecture가 구성된다. Progressive growing을 하여 마지막 층에선 고해상도의 이미지를 생성하게 된다. 여기서 정규화 방법으로는 AdaIN이 사용된다.

styleGAN2은 styleGAN의 AdaIN에 대한 문제점을 개선한 State of The Art 모델이다 . AdaIN이란 실제로 입력된 통계량을 사용하여 정규화하지만, 생성이미지 성능에 악영향을 미치는 droplet 현상을 야기한다. StyleGAN2에서는 실제 데이터의 통계량이 아닌 추정 통계량을 사용하여 convolution 가중치를 정규화하여 droplet현상을 방지한다.



그림a) 그림b)

스타일 블록의 내부 작업을 단순화했다. 이제 스타일 벡터에 의한 첫 번째 선형 변환(linear transformation)은 합성곱 연산(convoultion) 안에서 처리되어 수행된다. 스타일 블록에서, 스타일 벡터 W의 선형 변환(linear transformation)인 계수 y_s가 사용된다. 합성곱(convolution) 가중치 w_ijk와 함께 s를 곱한 콘텐츠 이미지를 처리하는 작업은콘텐츠 이미지를 가중치 w_ijk와 s의 곱과 합성곱 연산(convolving)하는 것과 같다. 그래서 이 작업은 다음과 같이 다시 쓰일 수 있다. ((b)의 Mod 연산이다.)

$$w'_{ijk} = s_i \cdot w_{ijk},$$

다음으로, 합성곱(convolution) 내부 처리에서 정규화(normalization) 작업(여기선 표준편차로만 나뉜다.)을 수행하는 것을 고려하자. 여기에서 입력이 표준 정규분포와 출력의 표준편차를 따르는 것으로 가정한다.

$$\sigma_j = \sqrt{\sum_{i,k} {w'_{ijk}}^2},$$

여기서 원하는 작업은 표준 편차의 역수로 출력을 곱하는 것이다. 가중치 w_ijk와 합성곱(convolution)의 곱셈에 표준 편차의 역수를 곱한 작업은 표준 편차의 역수에 곱한 가중치 w_ijk와 합성곱(convolution)하는 것과 같다. 따라서, 이 정규화(normalization) 작업은 다음과 같이 수행된다. ((b)의 Demod)

$$w''_{ijk} = w'_{ijk} \Big/ \sqrt{\sum_{i,k} {w'_{ijk}}^2 + \epsilon},$$

이에 의해, 스타일 블록의 작업 순서는 스타일에 의한 선형 변환(linear transformation) -> 합성곱(convolution) -> 출력 정규화(normalization)가 하나의 합성곱 과정(convolution process)으로 표현될 수 있다. 정규화(normalization) 부분은 출력이 정규 분포라 가정한 정규화(normalization) 과정이다. 다른 말로, 물방울을 발생시키는 실제 분포를 사용한 정규화(normalization)는 사용되지 않는다. 물방울은 이를 사용할 때 나오지 않는다.

그림) StyleGAN1의 물방울 현상

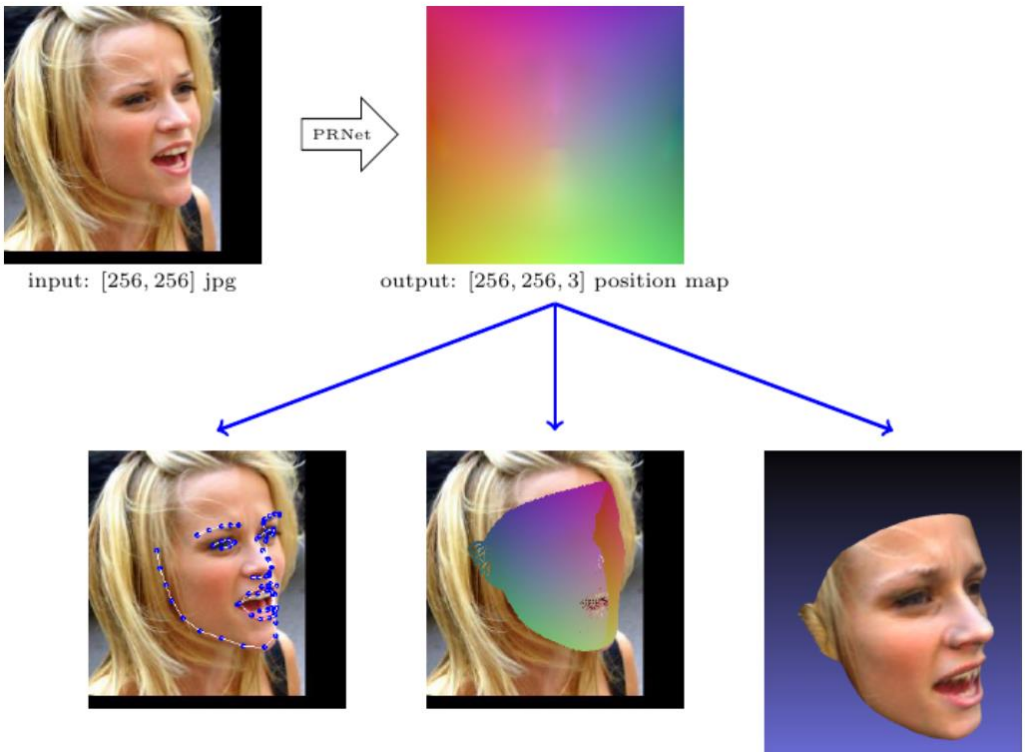
그림) StyleGAN2로 생성된이미지

우리는 이렇게 짜여진 algorithm으로 pretrained된 StyleGAN을 사용했고 이를 우리의 취지에 맞게 변형하여 사용했다. 300여개의 animated films의 캐릭터를 추가로 사용하여 학습시켰다. 또한 원래 실제 사람의 얼굴로 학습된 모델이기 때문에 Blending을 활용하여 fine tuning하였다. Blending이란 애니메이션 데이터셋과 실제 사람의 이목구비를 치환하여 새로운 dataset을 만드는 technique이다.
이러한 technique을 추가로 사용한다면 더 자연스러운 character output을 만들어낼 수 있다.

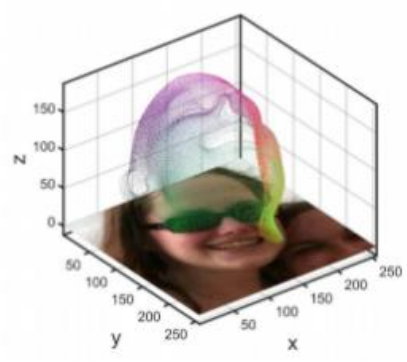
그림) 생성된 character 이미지

• PRNet

StyleGAN2 모델을 통해 원본 얼굴을 캐릭터화한 결과물에 PRNet을 적용한다. PRNet은 2D 얼굴의 특징을 추출하여 3D 얼굴로 reconstruction해주는 모델이다. 3D 얼굴을 만들기 위해 2D 얼굴 이미지에서 얼굴의 형태를 알려주는 landmark를 추출해 입체정보의 기반을 만든다. landmark를 이용해 추출한 얼굴의 형태는 x, y, z 좌표로 지정된다. 2D 얼굴 이미지를 이용해 3D 얼굴을 생성하는 모델은 3D vector data를 2D 이미지로 늘려 변환해야한다. PRNet 모델 역시 3D 얼굴 생성의 기반이 되는 3D vector를 늘린 2D 이미지에 UV position map을 이용해 해당 2D 이미지는 얼굴의 형태를 x, y, z 좌표로 지정한 것을 RGB 채널과 mapping하여 표현하는 것으로 구현되어있다. 이러한 방식을 통해 PRNet은 3D 얼굴을 생성하는 모델 중 input 2D 얼굴의 특징과 texture를 인종, 성별, 캐릭터에 가리지 않고 자연스럽게 생성해주는 모델이기 때문에 PRNet을 선택했다.



input 2D 얼굴을 바탕으로 UV position map을 채우는 과정이다. 얼굴의 형태를 landmark를 통해 추정하는 과정이 왼쪽 과정이다. 이를 통해 추출된 얼굴의 형태는 x, y, z 좌표로 지정되며 이 좌표를 RGB채널과 mapping하여 UV position map을 채운 형태가 가운데 과정이다. 최종적으로 UV position map을 통해 input 얼굴을 3D 얼굴로 만든 결과가 오른쪽이 된다.



3D 이미지 생성을 위해 얼굴 형태를 x, y, z 좌표로 지정한 것을 RGB 채널과 mapping하여 3D 공간에 표현한 그림이다.

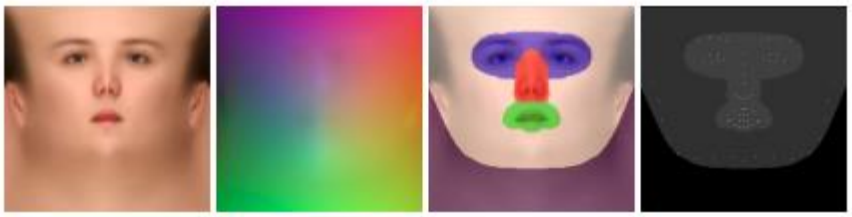


Fig. 4: The illustration of weight mask. From left to right: UV texture map, UV position map, colored texture map with segmentation information (blue for eye region, red for nose region, green for mouth region and purple for neck region), the final weight mask.

순서대로 UV texture map, UV position map, colored texture map, weight mask이다. 모든 map과 mask는 3D 얼굴 생성을 위해 필요한 부분을 모두 모아 2D 이미지로 늘린 것이다. UV texture map은 input 얼굴의 texture를 받아오기 위해 필요한 map이다. UV position map은 앞서서 설명했으며, colored texture map은 RGB 채널에 따라 가중치를 줄 부분을 나눠서 표현한 map이다. R 채널에서는 코 부분의 가중치가 높고, G 채널에서는 입 부분의 가중치가 높고, B 채널에서는 눈의 가중치가 높다. 이를 이용해 얼굴의 형태를 landmark를 통해 3D화 시킬 때, 가중치를 높게 줄 부분을 정할 수 있다. 눈, 코, 입 부분의 가중치가 더 높은 것을 마지막 weight mask에서 볼 수 있다.

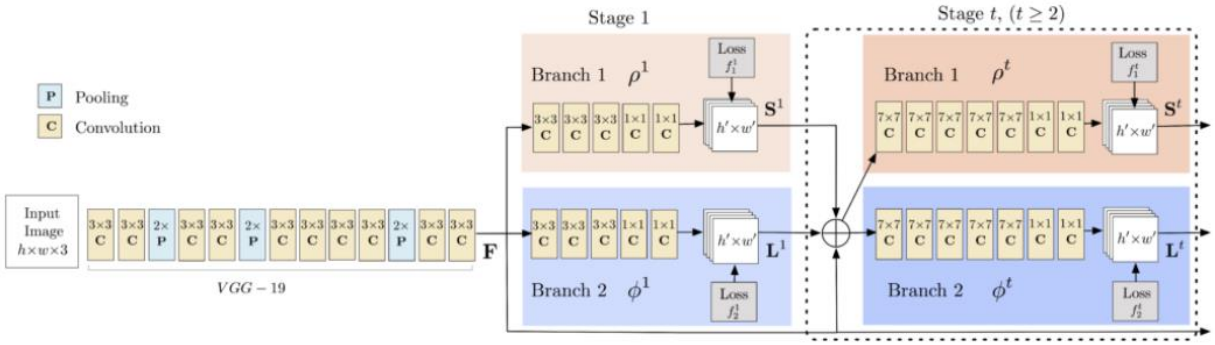
2.2.2. 캐릭터 움직임 생성

• Openpose

- 모델 설명

Openpose는 CNN 기반의 딥러닝 기술로 이미지 혹은 영상을 input으로 사람의 관절에 맞게 손, 발, 얼굴 등을 포인트로 하여 사람의 움직임을 감지하는 딥러닝 기술이다.

Caffe와 OpenCV 를 기반으로 구성하여 몸의 움직임을 추적하는 API 이고, 이에 따라 센서 없이 일반 카메라로도 이미지와 동영상 속 사람의 움직임을 추출 할 수 있다.



< Openpose 전체 CNN 모델 구조도 >

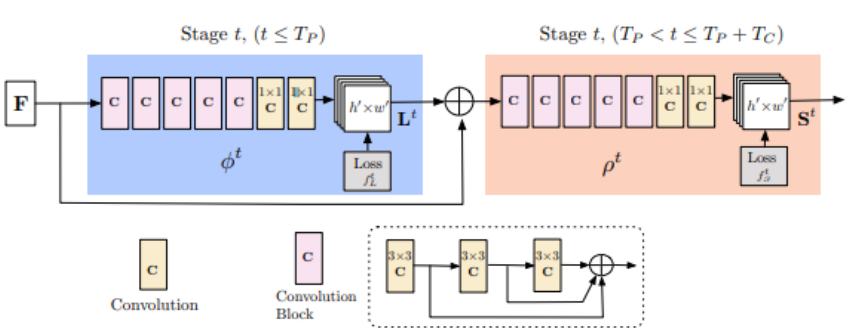
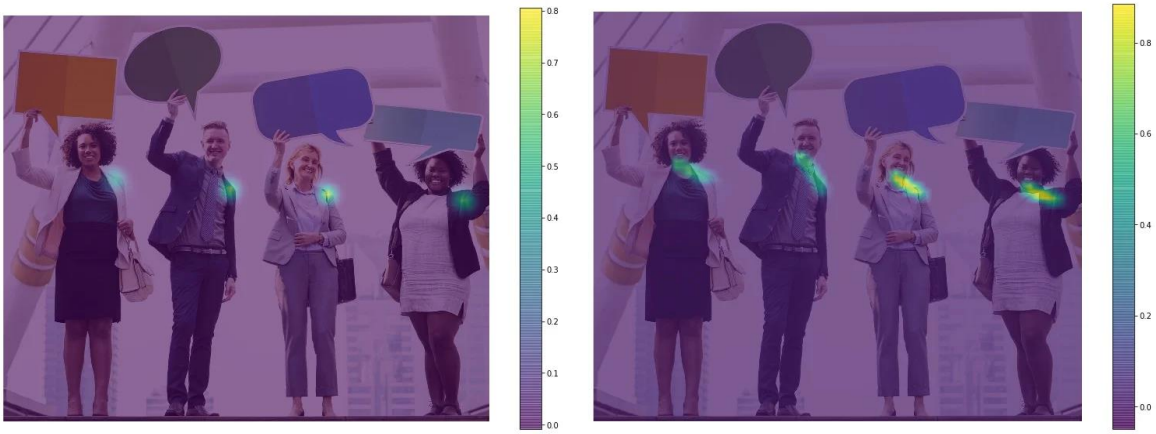


Fig. 3: Architecture of the multi-stage CNN. The first set of stages predicts PAFs $\mathbf{L}^t$, while the last set predicts confidence maps $\mathbf{S}^t$. The predictions of each stage and their corresponding image features are concatenated for each subsequent stage. Convolutions of kernel size 7 from the original approach [3] are replaced with 3 layers of convolutions of kernel 3 which are concatenated at their end.

< Openpose 다단계 CNN 구조의 첫 세트>

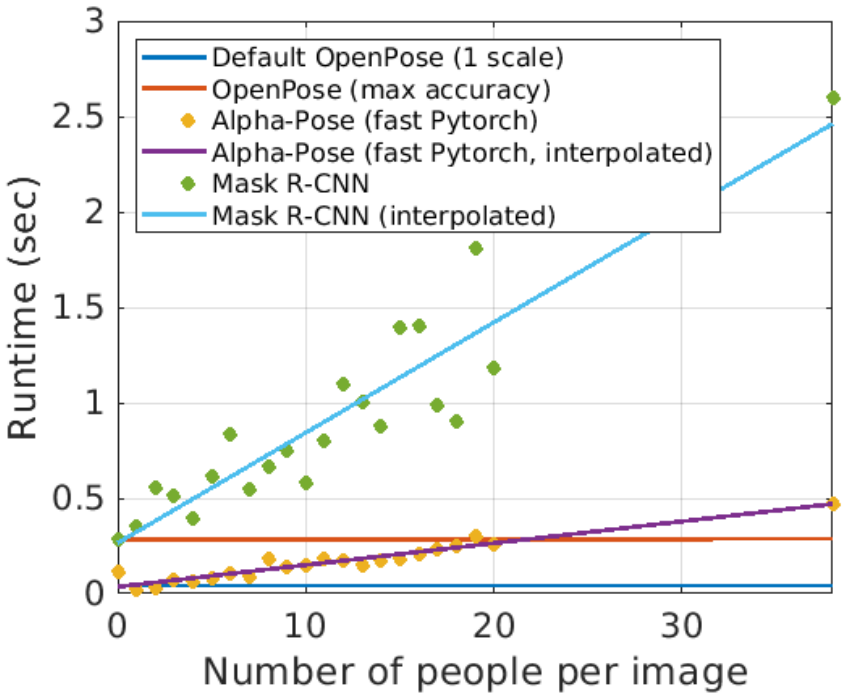
위의 전체 구조도에서 보는 것과 같이, 처음 입력된 데이터를 VGG Net -19 를 통해 수행된 Output 데이터의 특징, 즉 Feature을 강조한 상태로 출력하고, 이 출력된 Output을 두 번의 과정을 거쳐 전파를 수행하게 되는데, 첫번째 과정에서는 전반적인 신체 부위의 특정에 사용된다. 반복되는 stage를 따라 가지를 거쳐서 confidence map과 affinity field를 구하게 되는데, 여기서 confidence map은 인간의 관절 구조 등을 찾는데 사용되며, affinity field는 추출된 관절 구조가 어떤 객체의 것인가에 대해 알아내는데 사용된다.
좀 더 자세히 살펴보면, Openpose의 다단계 CNN 구조의 첫 세트 구조도에서 확인 할 수 있는 것과 같이, 스테이지의 첫번째 세트는 PAFs Lt 를 예측하고, 마지막 세트는 confidence map의 St 를 예측하며, 각 단계의 예측과 그에 해당하는 데이터의 특징 (feature)은 각 다음 단계에 연결되어 반복된다.



< 좌 : 인간의 왼쪽 어깨에 대한 confidence map 사진, 우 : 2D vector fields 예측 결과 사진 >

예를 들어, 인간의 왼쪽 어깨에 대한 confidence map은 위 사진의 왼쪽과 같이 형성되며, 그 후 지속적으로 다음 분기를 거쳐가며 인간의 각 부위 간 유사성에 기반하여 오른쪽 사진과 같이 2D vector fields를 예측하게 되고, 그 결과 왼쪽 어깨와 목덜미의 유사성을 표시한 그래프가 다음과 같이 나타난다. 이렇게 연산된 confidence map과 affinity field로 input에 존재하는 사람들의 keypoint를 예측하고 생성하며, 이 과정을 반복하여 학습하면 결과적으로 Openpose를 완성하게 된다.

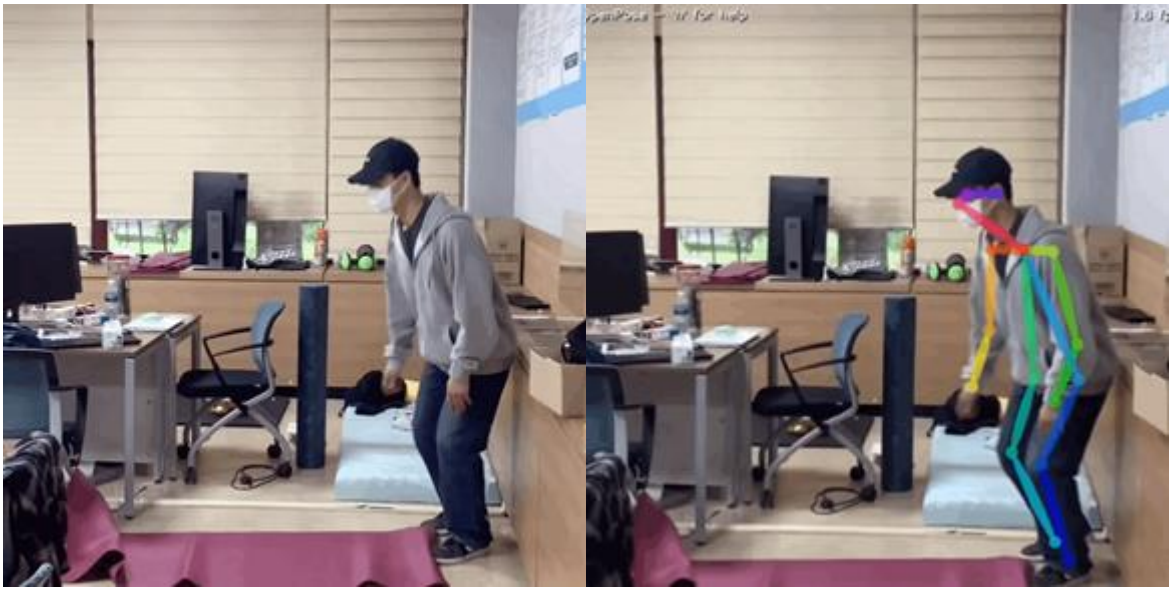
- 모델 선택



< Multi-person pose estimation model 간 성능 비교 >

Multi-person pose estimation model 중에서 가장 유명한 모델으로는 Openpose 와 Alphapose, Mask R-CNN 이 있다. 그 중에서 위의 성능 비교 그래프에서 알 수 있다시피, Openpose 모델이 사람 수가 늘어나도 실행속도가 낮아지지 않는 가장 안정적인 성능을 보이는 것을 알 수 있고, 이에 따라 본 프로젝트에서는 한 사람의 움직임만을 추출하여 상관없으나, 추후에 여러 사람으로 액션을 넣을 수도 있다는 점을 고려. 그리고 안정적인 모델을 선택하기 위해 Openpose 모델을 최종적으로 선택하였다.

다음은 우리 프로젝트에서 캐릭터에 움직임을 넣기 위해, 넣고자 하는 input 움직임을 영상으로 촬영 후, Openpose를 통해 관절 정보의 움직임을 얻어낸 모습이다.



< 프로젝트를 위해, 원하는 동작을 Openpose를 이용하여 움직임의 정보를 얻는 과정 >

• 3D-pose-baseline

- 모델 설명

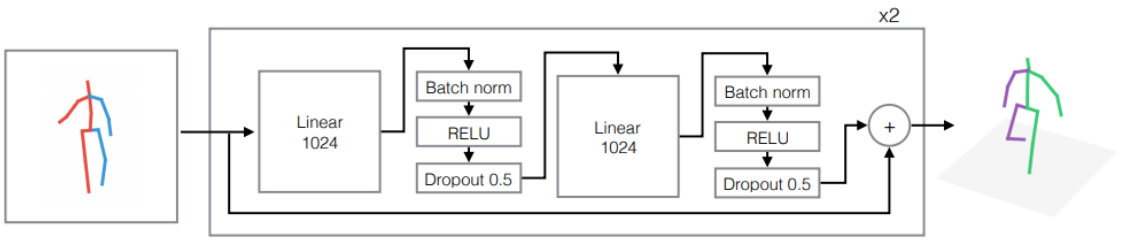


Figure 1. A diagram of our approach. The building block of our network is a linear layer, followed by batch normalization, dropout and a RELU activation. This is repeated twice, and the two blocks are wrapped in a residual connection. The outer block is repeated twice. The input to our system is an array of 2d joint positions, and the output is a series of joint positions in 3d.

< 3d-pose-baseline 전체 구조도 >

3D-pose-baseline 기술은 2D 관절의 위치를 3D 공간으로 옮겨 자세를 3D 인간 자세에 맞춰 더 분명하게 사람의 관절을 예측하는 기술이다. 위의 전체 구조도를 참고하여 살펴보면, 2D의 관절 정보를 Linear layer을 해주고, 이후 Batch normalization, dropout, RELU를 진행하고, 이를 두 번 반복한다. 그리고 이 두 번을 하나의 과정으로 묶은 후, 이 과정을 두 번을 다시 또

반복한다. 위 과정을 거치면서, input으로 들어온 2D의 연결 포지션들은 3D의 연결 포지션으로 나오게 된다.

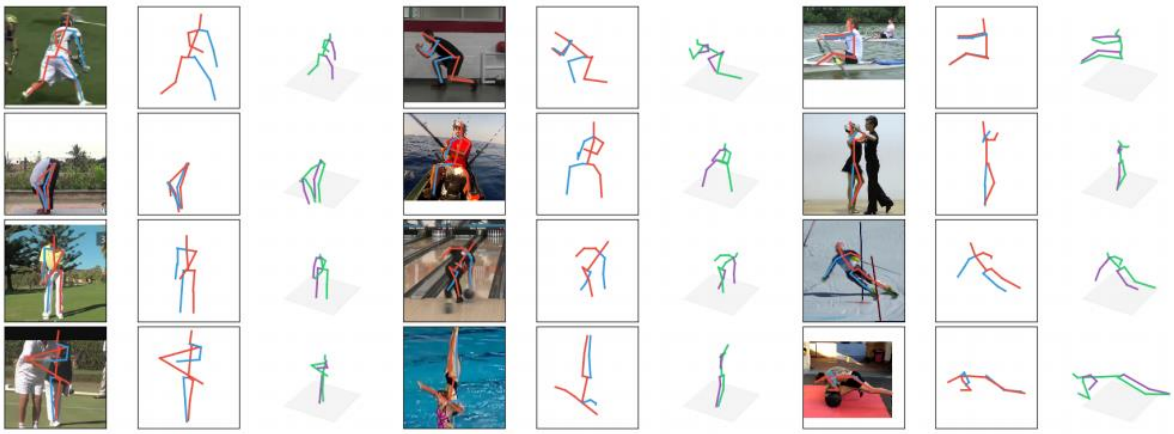


Figure 3. Qualitative results on the MPII test set. Observed image, 2d detection with Stacked Hourglass [30], (in green) our 3d prediction. The bottom 3 examples are typical failure cases, where either the 2d detector has failed badly (left), or slightly (right). In the middle, the 2d detector does a fine job, but the person is upside-down and Human3.6M does not provide any similar examples – the network still seems to predict an average pose.

< 3d-pose-baseline의 결과값>

이는 위의 사진과 같이, input의 이미지와 영상의 Openpose로 나온 움직임에 대한 골격 정보를, 3D 환경으로 옮겨 자세를 3D 인간 자세에 맞춰 보다 정확하고 분명하게 포즈를 예측하는 기술이다.

- 모델 선택

앞서 본 것과 같이 Openpose를 통해, 사람의 움직임을 가져오게 되면, 2D로 사람의 관절 위치와 그에 따른 움직임을 예측하기 때문에, 측면에서 촬영 할 경우 보이지 않는 손과 다리의 관절 정보가 예측되지 않는다던가, 프레임 밖으로 손이나 발과 같은 움직임 예측의 포인트 부분이 나가는 경우, 그에 대한 정보가 예측되지 않는 경우가 발생한다. 이 때, 이에 대한 오류를 줄이기 위해, 2D로 예측한 Openpose 모델 output 결과값을 다시 한 번 3D-pose-baseline 을 통해, 2D의 자세를 3D 공간으로 이동시켜, 3D 인간 자세에 맞춰 더 분명하게 움직임(관절 정보)을 예측하게 한다.

• FCRN (Fully Convolutional Residual Network)

- 모델 설명

FCRN은 Fully Convolutional Residual Network로, 고도화된 CNN 모델로 단일 RGB 이미지에 주어진 장면의 깊이 맵(Depth map)을 추정하는 모델이다. 간단하게는 depth camera를 구현했다고 생각하면 된다.

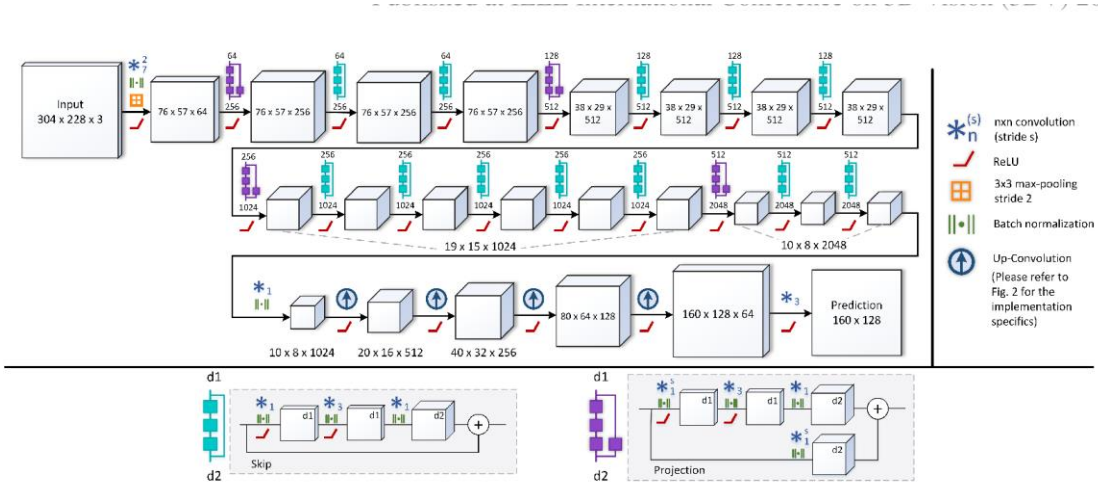
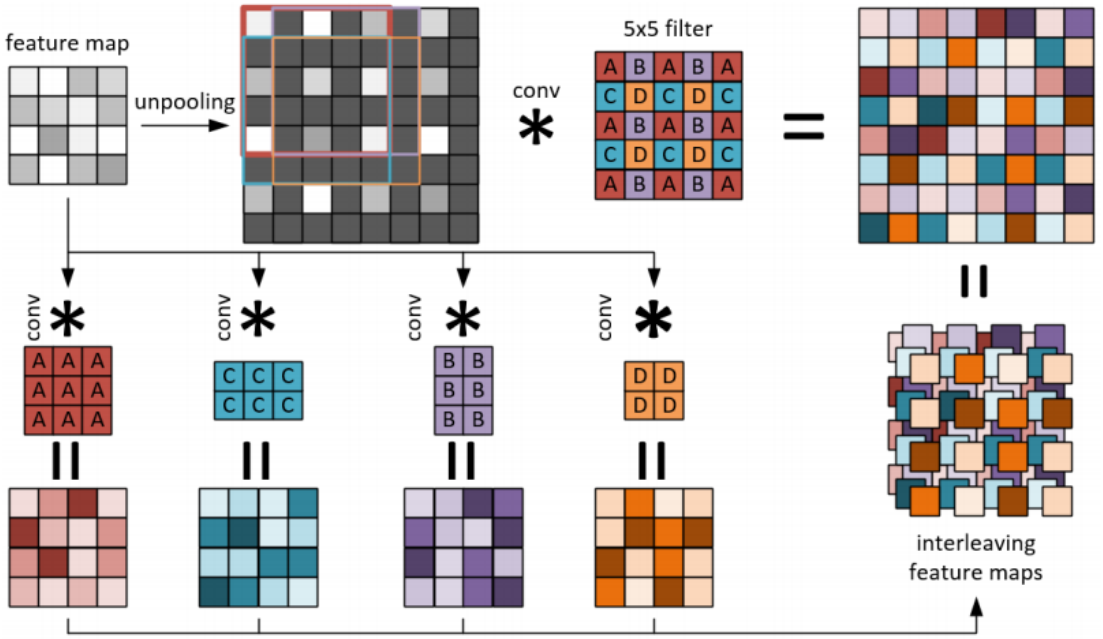


Figure 1. **Network architecture.** The proposed architecture builds upon ResNet-50. We replace the fully-connected layer, which was part of the original architecture, with our novel up-sampling blocks, yielding an output of roughly half the input resolution

< FCRN의 전체 구조도 >

위의 전체 구조도에서 보는 것과 같이, FCRN은 Resnet-50 을 기반으로 높은 레벨의 feature 와 낮은 레벨의 feature까지 전부 잡기 위해서, encoder과 decoder 아키텍처(architecture)를 사용하고, 이 때 encoder 부분은 Resnet-50 을 사용하고, decoder 부분은 up projections을 포함한다.



< FCRN의 빠른 up-convolutions >

또한 빠른 up-convolutions을 위해, 위의 빠른 up-convolutions 그림과 같은 과정을 지나게 되는데, 이는, 맨 윗 줄에서 보는 것과 같이, feature map의 size를 빈 칸은 0으로 채우는 unpooling을 두 번 거치게 되고, (4*4 > unpooling > 6*6 > unpooling > 8*8) 그 값을 A, B, C, D (위치만 표시하며, 실제 weight value는 전부 다르다.) 로만 이루어져 있는 5*5 convolution 필터를 0이 아닌 값들과 곱하는 과정으로 이루어져있다.

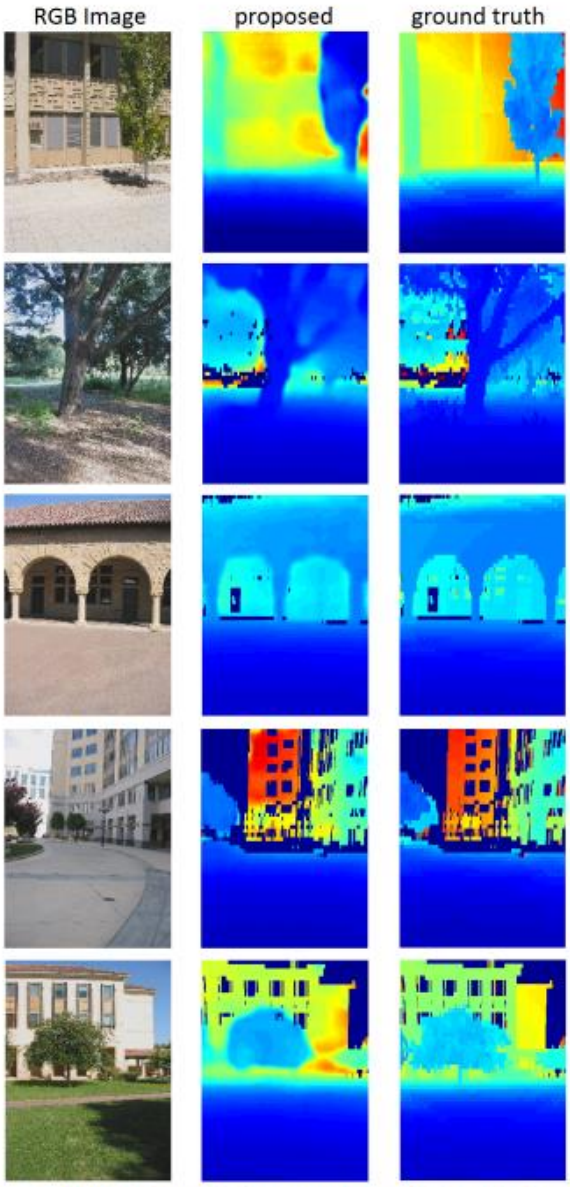
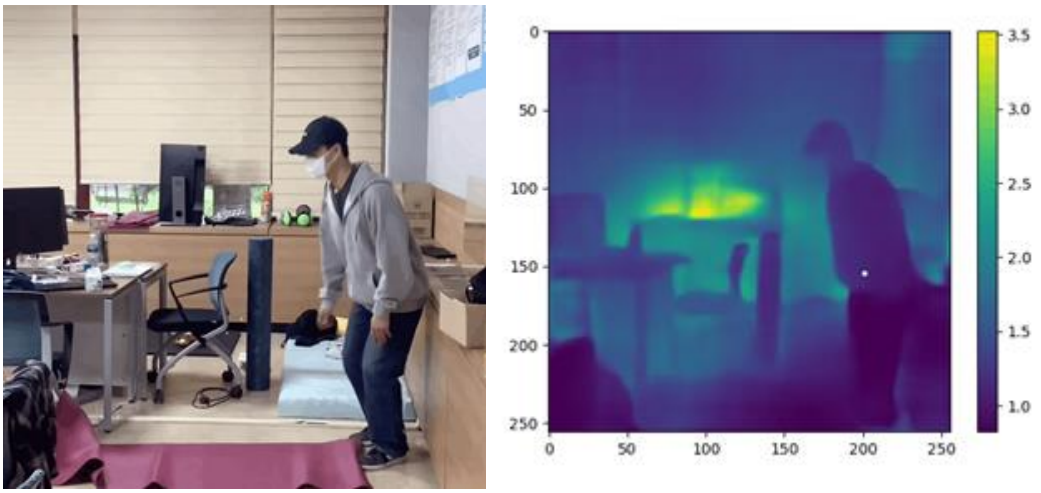


Figure 5. **Depth Prediction on Make3D.** Displayed are RGB images (first row), ground truth depth maps (middle row) and our predictions (last row). Pixels that correspond to distances $> 70m$ in the ground truth are masked out

< FCRN을 통한 결과 >

FCRN의 일련의 과정을 모두 마치면 다음 위의 사진과 같이, 결과를 얻게 되는데, 첫번째 열의 사진들은 실제 사진의 이미지, 두번째 열은 실제 Depth map 이며, 세번째 열은 FCRN을 통해 예측한 Depth map 이다. FCRN은 굉장히 고도화된 CNN 모델로 높은 정확도의 Depth map을 예측하는데, 이는 다음과 같이 이미지 속 오브젝트와 모델 등의 depth를 측정하고, 그 깊이를 색상의 차이로 표현한다. 간단하게 Depth camera의 성능을

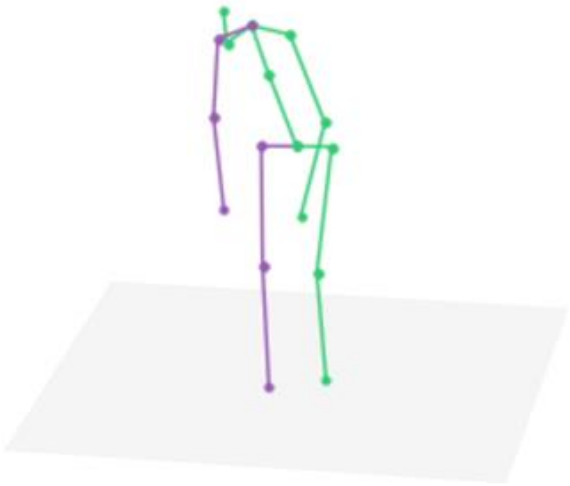
구현한다고 생각하면 되며, 값 비싸고, 무거우며, 구매 자체도 어려운 Depth camera의 성능을 CNN 모델을 통해 구축해낸 모델이다.
다음은 본 프로젝트에서, 움직임을 얻고자 하는 영상을 input으로 FCRN을 통해 output의 depth map을 얻은 결과이다.



< 프로젝트를 위해, 사용된 FCRN 모델의 input(좌)과 output(우) >

- 모델 선택 이유

앞서 설명한 것과 같이 Openpose를 통해, 사람의 움직임을 가져오게 되면, 2D로 사람의 관절 위치와 그에 따른 움직임을 예측하기 때문에, 측면에서 촬영 할 경우 보이지 않는 손과 다리의 관절 정보가 예측되지 않는다던가, 프레임 밖으로 손이나 발과 같은 움직임 예측의 포인트 부분이 나가는 경우가 생기는데, 그 때 그에 대한 정보가 예측되지 않는 오류가 발생하므로, 이를 FCRN 모델을 통해 이미지 속의 오브젝트와 모델 사이의 depth 차이 등 Depth map을 구현하여, 3D 구조를 정확히 예측하였다. 그리고 이를 3d-pose-baseline 으로 나온 모델의 3d 관절 정보 예측 output과 결합하여, 보이지 않고, 2D로 예측하여 한계가 있었던 Openpose 의 잘못 예측된 관절 정보를, input의 움직임에 대한 관절 정보를 정확히 예측하도록 보정하였다. 다음은 본 프로젝트에서 Openpose, 3D-pose-baseline, 그리고 FCRN 을 사용하여 움직임의 관절 정보를 예측한 Output이다.



< 본 프로젝트를 위해, 직접 도출한 움직임의 총 output >

2.3. AR

2.3.1. 캐릭터 몸체 생성

• Blender

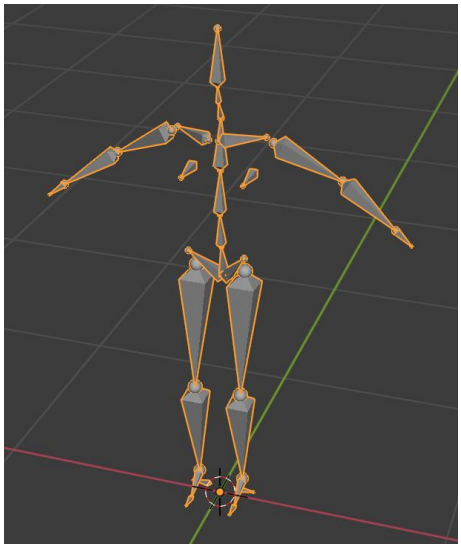
블렌더는 3D 컴퓨터 그래픽 제작 소프트웨어이다. 제품디자인, 게임 모델링, VFX아트, 애니메이션, 건축 등 다양한 분야에서 사용 가능하다. 가장 기본적인 모델링에서 부터 텍스처링, 그리기, 렌더링, 스컬프팅, 리깅, 애니메이팅 등 업계에서 필요로 하는 모든 기능을 지원하며 무료이다.
Autodesk사의 Maya 와 비교했을 때 가장 큰 장점은 역시나 무료라는 점이며, 사용자 커뮤니티가 활발하여 양질의 유튜브 강의를 쉽게 접할 수 있다. 그렇기 때문에 우리 프로젝트에서 3D캐릭터 생성을 위한 툴로 선정하게 되었다.

완성된 3D캐릭터 생성을 위한 과정은 다음과 같다. 먼저 StyleGaN2와 PRNET으로 생성해낸 캐릭터의 얼굴 파일(.obj)을 블렌더로 임포트 한다. 이 때 주의할 점은 PRNET에서 생성된 오브젝트 파일(.obj)과, 텍스처파일(.png)를 같은 폴더에 저장해야 한다는 것이다. 다른

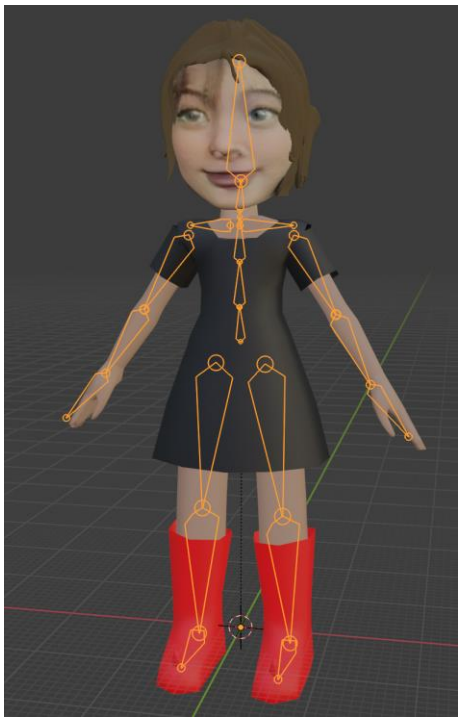
경로에 저장이 되어 있다면 블렌더에서 파일을 불러올 때 얼굴의 텍스처 정보가 입력 되지 않는다.

얼굴 파일을 불러왔으면, 사전에 탐색 혹은 제작해둔 머리와 몸통 오브젝트와 결합시킨다. 서로 다른 두가지 이상의 오브젝트를 결합할 때는 “join” 혹은 “parent”로 결합하게 되는데 “join” 기능을 사용할 때 얼굴 오브젝트를 마지막에 선택해주어야만 한다. 그러지 않으면 얼굴 정보가 모두 날아가버리는 문제가 생긴다.

얼굴, 머리, 몸통을 결합하였다면 사람의 움직임 정보를 입력하기 위한 사전작업으로 관절을 캐릭터 오브젝트에 연결하는 작업을 한다. 관절은 armature라는 항목을 추가하면 생성할 수 있는데 armature basic model 에서 기본 사람 모델, 동물 모델을 불러올 수 있으므로 일일이 single armature를 연결하는 것 보다 basic model을 활용하는 것이 효율적이다.

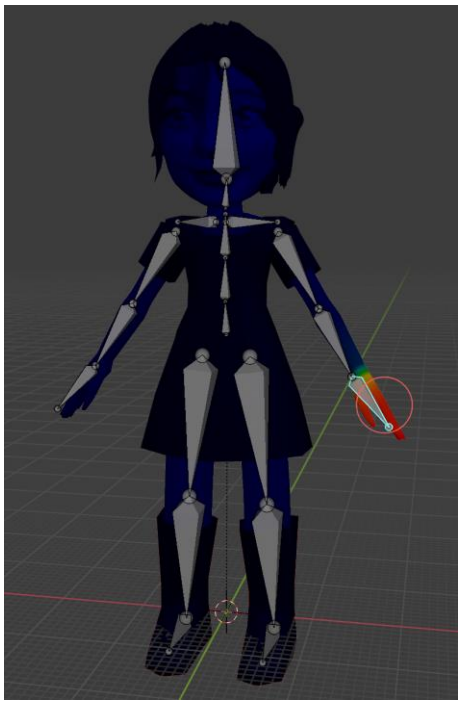


이렇게 관절을 가지고 왔다면 캐릭터 오브젝트와 위치정보를 일치시키고 parent → Armature deform → with Automatic weights 혹은 그냥 Armature deform을 선택하여 캐릭터와 관절을 연결시켜준다. 이 때 캐릭터를 먼저 선택 한 후 관절을 선택하여 parent로 묶어주어야만 한다.



armature와 캐릭터를 연동시켰다면 armature를 선택하고 “pose mode”로 들어가 관절을 움직여보며 잘 연결되었는지 확인한다. 잘 연결이 되어있지 않으면 weight paint 모드로

들어가 관절과 캐릭터의 가중치를 하나하나 색을 칠하여 입혀주어야 한다. weight paint 모드는 캐릭터를 먼저 선택하고 관절을 선택해야 접근 가능하다.



캐릭터와 관절이 제대로 연결 되었다면 OpenMMD 모델로 생성한 모션파일(.vmd)을 관절에 임포트 한다. 이 때 주의할 것은 vmd파일은 pmx파일로 불러온 오브젝트에만 움직임이 적용된다는 것이다. 만약 직접 작업한 관절정보에 vmd파일이 임포트 되지 않는다면, OpenMMD에서 제공하는 기본 캐릭터의 관절정보를 가지고와 내 캐릭터에 다시 입히는 작업을 해야하며, 정밀하게 작업을 해야 움직임이 무너지지 않는다.

캐릭터에 모션까지 모두 입혔으면 마지막으로 확장자명 gltf로 파일을 export 한다. gltf파일은 색정보, 움직임 정보를 모두 기억하고 있어 텍스처 깨짐 현상을 방지할 수 있다. export된 파일은 Reality Composer에 사용된다.

2.3.2. AR 환경 구축

- Unity

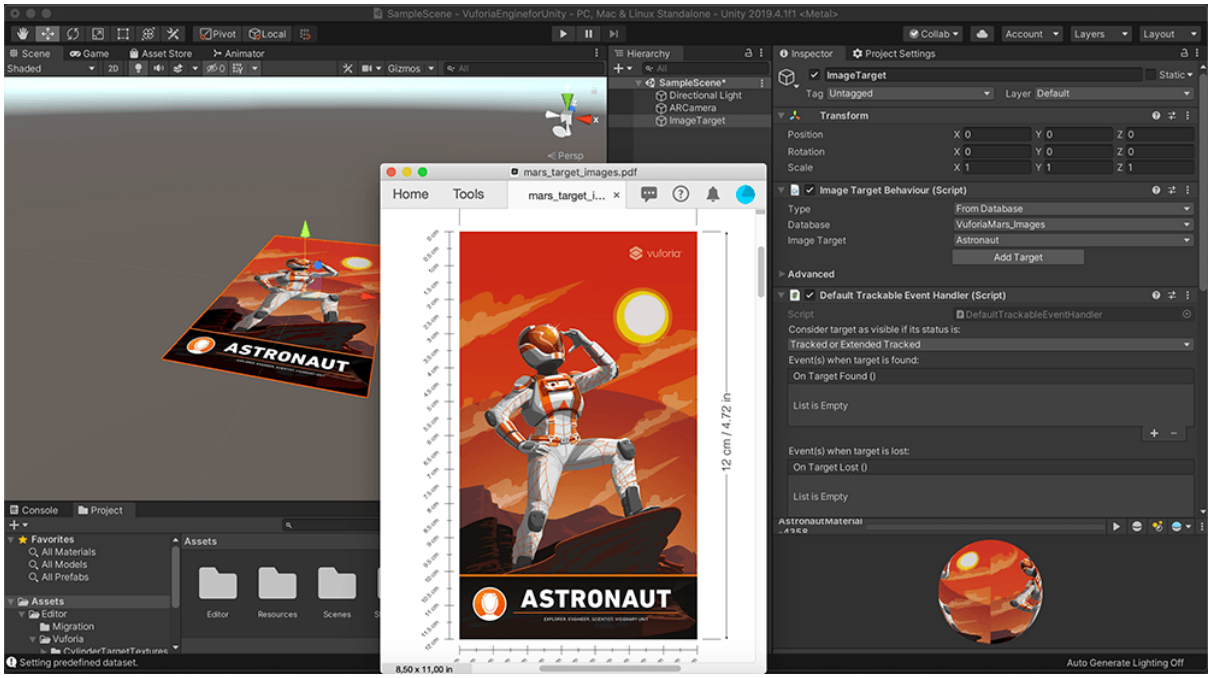
Unity는 3D 게임 개발 엔진으로, 2D 환경 뿐만 아니라, AR과 VR 등 다양한 3D 환경까지 구축 가능한 웹미디어 제작까지 가능하다. 대표적인 AR 게임 '포켓몬 고'를 실행하면 게임 첫 화면에 'Unity 3D'라는 로고를 볼 수 있다. 게임 뒷단에 유니티 기술이 사용되었다는 의미이다. '포켓몬 고' 뿐만 아니라 유니티는 현재 모바일 게임 분야에서 널리 사용되고 있다.

본 조에서는 생성한 3D 캐릭터를 세우기 위한 AR 환경 구축 도구로 유니티를 선택했다. 캐릭터는 obj 또는 fbx 파일 형식으로 export 할 수 있는데, 두 파일 형식 모두 유니티 엔진 내에 포함된 뷰포리아로 Import 가능하기 때문이다.

• Vuforia

Vuforia는 교차 플랫폼 증강 현실(AR) 및 혼합 현실(MR) 애플리케이션 개발 플랫폼으로 뷰포리아 SDK를 적용해 프로젝트에 AR 기능을 활성화 할 수 있다.

- Image Targeting 및 3D Object Positioning



뷰포리아 엔진을 통해 Image Target과 3D Object Positioning이 가능하다. 이 기능을 사용하여 제작한 3D 캐릭터 및 게임에 필요한 오브젝트들(황금열쇠, 무인도 등)이 특정 이미지를 인식했을 때 생성될 수 있는 환경을 만들고자 했다.

먼저 새로운 Scene을 생성한 후 AR 카메라를 적용했다. 뷰포리아 홈페이지에서 App License key를 만들어 적용하여 AR 카메라 기능을 활성화 했다. 이 카메라를 통해 저장한 이미지를 비추면 이미지 위에 설정한 오브젝트가 생성될 수 있도록 했다.

아래 그림은 데이터베이스에 이미지를 미리 입력한 후, 해당 이미지를 AR 카메라에 비추었을 때 설정한 3D 오브젝트가 생성되는 모습이다. 유니티와 뷰포리아 엔진을 사용했을 경우 2개의 모델까지는 멀티타겟팅이 잘 이루어지나, 3개 이상의 모델부터는 멀티 타겟팅이 잘 구현되지 않는 문제점이 있었다.

위 문제를 해결하기 위해 새로운 플랫폼 'Reality Composer'를 사용했다.

• Reality Composer

Reality Composer는 apple사에서 지원하는 어플리케이션으로 기존 3D 모델을 USDZ로 변환하여 apple 도구와 AR을 지원하는 모든 apple 기기에서 사용 가능하다. 내장된 AR 라이브러리에 USDZ 파일을 사용하거나 자체 USDZ 파일을 가져올 수 있는데, 본 조에서는 앞서 생성한 3D 캐릭터를 USDZ 파일로 가져와 AR 환경에 세웠고, USDZ 애니메이션을 추가했다.

게임 보드의 이름은 'PosMarble'로, 포스코 AI·Big Data 아카데미의 생활을 반영했다. 각 칸마다 '워크스테이션 고장', '노벨이 쓰다듬고 오기', '빅데이터 발표 전 날로 돌아가기' 등 아카데미 생활 중 일어날 수 있는 이슈 또는 수행해야 하는 미션을 넣어 교육생들의 3D 캐릭터와 함께 수료를 향한 여정을 떠나는 게임을 즐길 수 있는 환경을 구축했다.

<AR환경에서 본 3D ‘Posmarble’ 맵>

3. 기대효과 및 개선기회

3.1. 기대효과

• 친목 공간 사이버로의 확대

AR 보드게임을 나만의 캐릭터와 함께 즐기는 현 프로젝트는 MZ세대가 자신을 표현하는 공간이 되는 것과 더불어 코로나19 시국에 안전하게 주변인 또는 불특정 타인과 친목을 다질 수 있는 수단으로 사용될 수 있다고 예상한다.

• 교육용 콘텐츠

현재, 메타버스를 접목한 키즈 콘텐츠가 증가하고 있는 추세이므로 사용자 커스텀 캐릭터, 커스텀 스토리 제작, 그리고 AR을 통한 시각 효과를 모두 누릴 수 있는 현 프로젝트를 키즈 교육용 콘텐츠에 접목시킬 수 있다고 예상한다. 이렇게 접목한 교육용 콘텐츠는 아이들에게 자신만의 캐릭터를 생성하고, 스스로 스토리를 제작하게 할 수 있어 자신만의 선호와 주관을 정립하는 것에 도움을 주는 것과 더불어 아이 주도형 놀이공간으로 공부에 대한 흥미를 유발시킬 수 있다고 예상한다.

3.2. 개선기회

현 프로젝트에서 딥러닝모델을 통해 3D 캐릭터를 생성할 때, 캐릭터의 얼굴만 나오므로 blender 툴을 이용해 직접 생성된 얼굴과 머리, 몸체를 결합해야했다. 또한 AR 환경 구축을 위해 사용한 툴인 reality composer에서 자신이 생성한 custom 보드판이 타인과 연동이 잘 되지 않는 문제가 있었다.

그래서 이를 자체적인 AR 게임 어플리케이션 개발을 통해 자동으로 자신의 캐릭터를 생성해주는 시스템을 구축하면 자동으로 자신의 캐릭터를 생성해주는 것과 더불어 실시간으로 멀티유저와 AR 보드게임 플레이가 가능할 것으로 예상한다.

4. 참고문헌

- Deeper Depth Prediction with Fully Convolutional Residual Networks (Proposed by Iro Laina and Christian Rupprecht at the IEEE International Conference on 3D Vision 2016.)
- FCRN-DepthPrediction _ github
- OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields (Proposed by Gines Hidalgo, Zhe Cao, Tomas Simon, Shih-En Wei, Hanbyul Joo, and Yaser Sheikh at CVPR 2017.)
- Openpose, CMU-Perceptual-Computing-Lab _ github
- A simple yet effective baseline for 3d human pose estimation(Proposed by Julieta Martinez, Rayat Hossain, Javier Romero, James J. Little In ICCV, 2017)
- 3d-pose-baseline, una-dinosauria _ github
- Lifting from the Deep: Convolutional 3D Pose Estimation from a Single Image (Proposed by Denis Tome, Chris Russell and Lourdes Agapito at CVPR 2017)
- OpemMMD, peterljq _ github
- Human-action-classification, dronefreak _ github
- Lifting-from-the-Deep-release, DenisTome _ github

- Analyzing and Improving the Image Quality of StyleGAN (Proposed by Tero Karras, Samuli Laine Miika Aittala, Janne Hellsten, Jaakko Lehtinen and Timo Alia at NVIDIA, 2020)
- Stylegan2, NVlabs _ github
- Stylegan2-pytorch, rosinality _ github
- Stylegan2-ada-pytorch, NVlabs _ github
- PRNet: Self-Supervised Learning for Partial-to-Partial Registration (Proposed by Yue Wang and Justin Solomon at ML, 2019)
- PRNet, YadiraF _ github

5. 코드

https://github.com/AI-14-A2/AI_14_A2