Pusan National University
Computer Science and Engineering
Technical Report 2022-16

스마트폰을 이용한 HMD 사용자의 가상환경 상호작용 몰입 개선 방법 연구

201924401 강경찬
201924472 박예린
201624456 김진서

지도교수 : 이 명 호 교수님

목 차

1. 서론

1.1. 연구 배경

코로나 이후부터 지금까지 메타버스에 대한 관심이 날로 증가하고 있습니다. 이에 따라 VR / AR 또한 각광받고 있습니다. 사람들은 가상공간에서 자신의 아바타를 통해 타인과 소통하고 다양한 활동을 합니다. 이러한 활동들을 하기 위해선 현실의 "나"와 가상공간의 "아바타"가 서로 상호작용을 통해 영향을 줄 수 있어야 합니다.

현재 VR / AR에서의 상호작용은 주로 VR Controller 혹은 Hand Tracking으로 이루어지고 있습니다. VR Controller는 정확하고 빠른 Tracking 기능을 제공하지만, 사용자가 할 수 있는 상호작용에 제한이 있습니다. Hand Tracking은 별도의 Controller가 필요 없다는 장점과 양손이 자유롭다는 장점이 있지만, 정확도가 부족하며 할 수 있는 상호작용이 VR Controller보다 제한적이라는 단점이 있습니다. 따라서 저희는 사용자들이 보다 많은 상호작용이 가능하게끔, 스마트폰을 이용한 VR/AR 상호작용을 제안합니다.

스마트폰을 이용할 경우 사용자는 별도의 VR Controller를 가지고 있을 필요도 없으며 한 손이 자유로워질 수 있습니다. 스마트폰의 다양한 센서 및 터치스크린을 이용해서 Hand Tracking에서의 제한적인 상호작용 문제를 해결할 수 있을 것이라고 저희는 기대합니다.

1.2. 기존 문제점

1.2.1. 기존 연구 사항

Face Tracking을 통해 현실 사람의 표정을 가상공간의 캐릭터로 표현하는 기술은 현존하나 HMD를 착용한 상태의 Face를 Tracking한 연구 결과는 많지 않습니다. 기존의 Face Tracking 기술은 HMD 기기를 착용하면 눈을 포함한 얼굴의 윗부분이 기기에 의해 가려지기에 HMD 기기를 착용한 사람의 Face Tracking에 바로 적용하기에는 힘이 듭니다.

스마트폰 카메라를 사용하는 본 과제와 달리, 기존 Tracking 연구는 VR 기기에 딸린 Camera 또는 Depth Camera로 Lip Motion Tracking 또는 Object Tracking을 하였으나 이는 일반 사용자들 입장에서는 접근성이 떨어집니다.

1.2.2. 작년 졸업과제의 한계점

작년 이명호 교수님 연구실에서 진행된 스마트폰을 이용한 몰입형 가상현실 상호작용 기법 개발이라는 저희의 선행 연구가 존재합니다. 해당 과제에서는 스마트폰을 이용해 VR에서 사용 가능한 다양한 기능을 구현하였습니다.

그러나 실제 사용에 있어서는 많은 제약이 존재하였습니다. 스마트폰의 위치가 고정되어 있어야 하였고, 스마트폰의 위치가 바뀌게 되면 VR안에서 UI가 함께 움직이지 않는 등의 제약이 존재하였습니다. 따라서 본 과제에서는 스마트폰의 위치가 고정되어 있지 않아도 되도록, HMD Face Tracking을 통해 스마트폰 카메라의 위치를 보다 정확하게 추정하여 VR내의 UI가 함께 움직일 수 있도록 할 것입니다.

1.3. 연구 목표

1.3.1 HMD Tracking

- Oculus Quest2의 규격을 통해 CAD로 기기 modeling
- HMD 기기 위치 Tracking 정보를 바탕으로 VR 환경 스마트폰 위치 추정

1.3.2 Lip motion Tracking

- 딥러닝을 통해 HMD를 착용한 상태의 세 단계로 나누어 Lip motion을 검출 (closed, semi-open, wide-open)
- 검출한 class 중 가장 높은 신뢰도를 보이는 class를 통신으로 넘겨주어 VR 환경에 입술 모양 움직임 구현

2. 연구 배경

2.1. HMD Tracking

이번 연구에서 HMD Tracking을 진행하게 된 큰 이유는 Face Tracking을 통해 사람의 얼굴 표정 변화를 가상공간 상에서의 캐릭터의 얼굴에 보이게 하는 연구는 많이 있지만, HMD 기기를 착용한 사람의 얼굴은 눈과 코가 보이지 않기 때문에 얼굴 표정 변화를 바로 알 수 없습니다. 또한 많은 Tracking 연구들이 스마트폰 카메라가 아닌 oculus quest 2 에 장착된 카메라를 사용하거나 별도의 외부 카메라를 활용하여 Tracking이 진행되기에 많은 사람들이 카메라로 Tracking 기술을 활용할 수 있는 기회가 적습니다.

VR 공간에서는 사용자가 가상현실에 몰입을 하는 것이 중요합니다. 이러한 이유로 HMD를 쓰고 있는 사용자의 몰입을 높이기 위해 사용자가 가상현실에 둘러싸여 있는 느낌을 받게 하는 환경 Immersive Virtual Environment(이하 IVE)를 조성했습니다. IVE를 활용하여 가상현실의 몰입을 높이는 것은 좋으나, 사용자가 현실의 정보를 놓치게 되는 경우가 많아 아쉬운 점이 있었습니다. 이러한 문제점을 보완하고자 작년 졸업과제에서는 스마트폰 후면 카메라를 이용한 것과 반대로 이번 연구는 스마트폰의 전면 카메라를 활용하여 현실 환경을 보는 것을 바탕으로 HMD Tracking을 진행했습니다.

2.2. Lip motion Tracking

VR 환경에서 Virtual 아바타는 또 다른 나입니다. 만약 사용자의 음성은 나오는데 아바타의 입이 움직이지 않는다면 대화 참여자들은 실제로 대화하는 느낌이 들지 않습니다. VR 개발자의 가장 중요한 목표 중 하나는 사용자가 가상현실에 몰입하여 실제처럼 받아들이게 하는 것입니다. Lip motion Tracking을 통해 실사용자의 입 모양과 캐릭터의 입 모양이 상호작용을 통해 연동되도록 함으로써 "함께 소통하고 있다"고 사용자들이 느끼도록 하고자 합니다.

선행 연구 격인 전년도 졸업과제에서는 스마트폰이 VR 컨트롤러를 대체하거나 뛰어넘을 수 있는 가능성을 보여주었습니다. 본 연구에서는 스마트폰 카메라를 이용한 HMD Tracking을 통해 전년도 졸업과제의 문제점을 보완하고 Lip Motion Tracking을 통해 VR/AR 환경에서 보다 몰입 가능한 사용자 경험을 제공하고자 합니다. VR 기기 착용자의 Lip motion Tracking 은 많이 연구되었고 이미 시판된 기술이지만 페이스 X 트래킹을 지원하는 아이폰이 아닌 안드로이드 스마트폰을 활용한 경우는 드뭅니다. 하나의 안드로이드 스마트폰만으로, 컨트롤러의 기능과 Lip motion Tracking을 동시에 수행할 수 있는 기능을 저희는 제공하고자 합니다.

3. 연구 내용

3.1. TCP 통신

Python Server의 경우 스마트폰으로부터 이미지 파일을 받기 때문에 전송 속도와 정확도가 중요합니다. 따라서 TCP 통신의 특징인 이미지 파일의 byte 크기를 먼저 받고 해당 byte크기만큼 데이터를 받습니다. byte 크기가 경험적으로 10만이 넘어가는 경우가 많기에 한 번에 다 받지 못하는 경우가 종종있어 이미지가 깨지는 경우가 발생하고 이로 인해 다음 파일의 크기를 받아야 할 때 이전에 다 받지 못한 이미지 파일에 대한 정보가 받아지면서 연속적으로 에러가 생기는 문제가 생겼습니다. 그 문제를 해결하기 위해, 반복문을 사용해 이미지 파일의 크기만큼 다 받도록 하였습니다. PIL 라이브러리를 사용하여 byte array를 이미지로 변환시키고 Deep Learning에 사용할 수 있도록 yolov7-main 안의 image 폴더 안에 test.jpg로 저장하도록 하였습니다.

Python Client의 경우 VR의 ip 주소를 직접 적고 port는 50001로 고정하였습니다. Client와 딥러닝 코드가 별개로 있어서 딥러닝 코드에서 Client 코드 안의 함수를 불러오는 방식이었는데, 이 방법은 Client가 Connect 시도를 계속하여서 VR에 계속해서 정보를 줄 수가 없다는 것을 알게 되어 하나의 파일로 합치게 되었습니다. 딥러닝을 거친 결과값을 closed면 1, semiOpenMouth면 2, wideOpenMouth면 3을 보내도록 하였습니다.

```
PS C:\Users\kangk\OneDrive\Desktop\Graduation\PythonCommunication>
angk/OneDrive/Desktop/Graduation/PythonCommunication/server/TCP_se
Server socket [ TCP_IP: 192.168.0.29, TCP_PORT: 50000 ] is open
```

TCP_server.py 연결 전

```
Server socket [ TCP_IP: 192.168.0.29, TCP_PORT: 50000 ] is open
Server socket [ TCP_IP: 192.168.0.29, TCP_PORT: 50000 ] is connected with client
length : 45379
count: 45379 length: 45379
length : 44407
count: 44407 length: 44407
length : 52064
```

TCP_server 연결 후

python은 서버의 경우 위 사진과 같이 실행했을 경우 Server가 open되었다는 메시지와 함께 server가 열리고 스마트폰 client가 연결을 시도했을 때 connected with client라는 메시지로 연결에 성공했음을 알려줍니다.

```
Namespace(weights=['runs_fix/train/exp/weights/be
c_nms=False, augment=False, update=False, project
서버 (192.168.0.34:50001)에 연결 되었습니다.
YOLOR  34686d8 torch 1.12.1+cpu CPU
```

Modified_detect.py 실행 시

client의 경우 VR server가 켜져 있을 때 실행했을 경우 연결되었다는 메시지를 출력하여 연결에 성공했음을 알려줍니다.

VR 환경에서는 총 두 개의 Server가 필요합니다. Python으로부터 정보를 수신할 Server의 port는 50001로, 스마트폰 앱으로부터 정보를 수신할 Server의 port는 50002로 고정하였습니다. Server는 Client로부터 데이터 송신을 기다리므로 그동안 다른 코드가 실행되지 못하는 일이 벌어지지 않게 Thread를 사용하였습니다. 데이터의 길이를 먼저 받은 후, 다음 데이터를 받아옵니다.

통신 연결 전 VR 화면

통신 연결 후 VR 화면

HMD에서 앱을 실행하면 위 사진처럼 UI에 AR waiting과 python waiting이

라는 문구를 볼 수 있습니다. 그리고 밑에 현재 VR 기기의 ip주소도 적혀 있습니다. 스마트폰이 연결을 시도하여 성공하면 AR Connected로 바뀌고 python에서 연결을 시도하여 성공하면 python Client Connected라는 문구로 바뀝니다. 오른쪽에 FPS는 스마트폰으로 전송받는 속도를 나타냅니다. HMD tracking이 끊겨서 정보를 주고 있지 않을 때에는 FPS가 0임을 알 수 있습니다.

3.2. HMD Tracking

3.2.1. 스마트폰의 좌표 계산

HMD Tracking을 하는 주된 목적은 VR 환경에서 스마트폰의 실제 위치를 반영하여 확인할 수 있게 하기 위해서입니다. 실제 카메라 화면에 Database에 등록한 model이 감지될 경우 unity상의 Vuforia ARcamera Object와 실제 카메라가 겹쳐서 model의 위치에 따라 unity modelTarget object가 같이 움직입니다. 감지한 물체가 카메라 상에서 위로 올라갈 경우 unity modelTarget도 같은 거리만큼 위로 올라가고, 회전상태도 동일하게 적용됩니다. 따라서 unity ARcamera object와 modelTarget의 position과 rotation을 통해 HMD 기준 스마트폰의 position과 rotation을 알 수 있습니다.

V_A를 ARcamera의 Position Vector라고 하고, V_m를 modelTarget의 Position Vector라고하고 V_t를 TCP통신으로 보낼 벡터라고 할 때, $V_A - V_m = V_t$ 인 V_t에 대해서 VR 안에서 위치해야 할 스마트폰 object의 position Vector를 V_S , HMD의 position Vector를 V_H 라고 하면 $V_S - V_H = V_t$ 이므로 $V_S = V_H + V_t$ 임을 알 수 있습니다. VR 안의 스마트폰의 Position Vector에 이 공식을 적용하였습니다. Rotation Vector는 modelTarget의 Rotation Vector를 그대로 적용하였습니다.

3.2.2. shader 편집

Vuforia에서는 구 버전에서만 스마트폰 전면 카메라 접근이 가능했습니다. 기존에는 videoBackground라는 Vuforia에서 제공해주는 기본 shader를 사용하고 있었습니다만, unity에서 제공하는 shader인 ARCoreBackground를 적용해보았더니 전면카메라 접근이 가능했습니다. 하지만 후면 카메라로 Vuforia를 사용할 때보다 정확도가 떨어졌고, 휴대폰 화면의 Full Screen으로 나타나다보니 이미지가 찌그러져 보였으며 상하반전이 된 상태였습니다. 따라서 이 문제를 해결하기 위한 방안 세 가지를 생각하였습니다.

첫 번째는 기존 shader를 사용하며 전면 카메라에 접근하기 위해 external device에 대한 custom driver를 연구하는 것입니다. 두 번째는 Vuforia 구 버전을 사용하여 전면 카메라에 접근하는 것입니다. 세 번째는 shader를 편집하

여 촬영 화면을 조정해보는 것입니다. 첫 번째 방법의 경우 얻을 수 있는 정보가 적고 난이도가 너무 어려워서 보류하였습니다. 두 번째 방법의 경우 구 버전을 사용하는 것은 향후 발전 가능성이 있는 현 프로젝트에게 어울리지 않는 방법이라고 생각하여서 세 번째 방법을 택하게 되었습니다. 그 중에서도 ARCoreBackground Shader가 고칠 수 있는 코드들로 구성되어 있어서 이 방법을 사용하기로 하였습니다.

상하 반전 적용 전

상하 반전 적용 후

실험을 통하여 Texture Coordinate의 x좌표가 세로고 y가 가로임을 알게 되었습니다. 상하를 바꾸기 위해서 x좌표를 1-x로 만들었습니다. 기존에 1-y로 되어있던 부분을 y로 고쳐주었습니다. y의 경우 스마트폰을 가로로 눕혔을 때 상하 반전이 생기는 것을 확인하였고 1-y를 y로 고침으로써 그 문제도 해결되었습니다.

비율 조정 전

비율 조정 후

카메라 화면 비율을 조정하기 전에는 2080*1080 Full Screen의 비율로 보이기 때문에 기존의 자연스러운 4:3 비율에 비해서 세로로 길쭉하게 보입니다. 이렇게 비율 문제로 인해 찌그러져 보이는 현상은 HMD Tracking에 어려움이 생길 수 있기 때문에 shader의 코드를 수정하여 해결하려고 하였습니다. 이는 상하 반전과 마찬가지로 textureCoordinate을 수정하면 되는데 3.2.3에서 설명했듯이 가로를 나타내는 y에 1080/1560을 곱하면 됩니다.

y좌표 이동 전　　y좌표 이동 후

왼쪽 그림은 비율 조정만 한 사진에서 4:3 비율을 가정하였을 때 사라진 부분을 점선 사각형으로 채운 것입니다. 점선 사각형의 가로 길이는 1560에서 1080을 뺀 480이고 잘려나간 부분을 왼쪽이 아닌 양옆으로 반으로 나누어져 있게 하기 위해서는 textureCoordinate의 y좌표를 이동 시키면 됩니다. 따라서 비율 조정 전을 기준으로 480의 절반인 240에서 가로 길이인 1080을 나눈 만큼 더하면 되는데, 비율 조정을 하게 되면 1080/1560을 곱하게 되므로 240/1560만큼 y에 더하면 됩니다.

3.2.3. 스마트폰 위치 변경

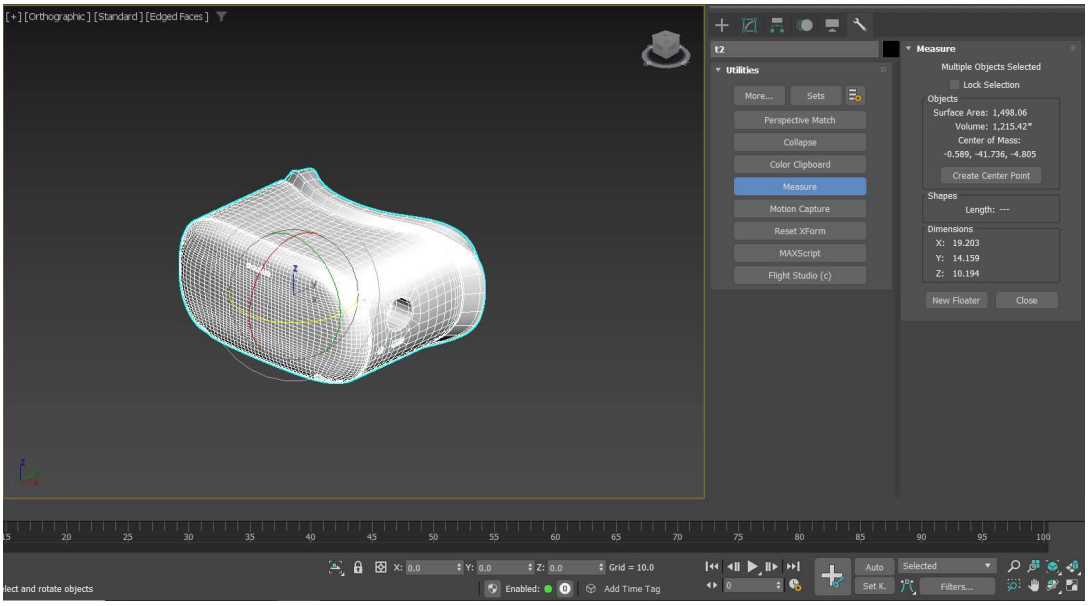
위에서 언급한 것처럼 ARcamera와 modelTarget의 position Vector 차이를 HMD camera의 position Vector에 더하면 VR 내에서 스마트폰의 position Vector임을 알 수 있습니다. 실제로도 오른쪽으로 움직이면 VR내 스마트폰이 오른쪽으로, 왼쪽이면 왼쪽으로 잘 반영됨을 볼 수 있지만 스마트폰의 위치가 스마트폰을 잡고 있는 손에 잘 위치하지 않는다는 어려움이 있었습니다. 그래서 Oculus에서 제공하는 hand tracking이 되고 있는 손 object의 좌표를 이용하여 가장 바람직한 곳에 존재하는 스마트폰의 위치 Vector를 InHandPhoneVector로 저장해두고 스마트폰의 위치와 InHandPhoneVector의 거리가 일정 이상 벌어지

면 그 차이를 reviseVector로 저장하고 그 벡터를 HMD tracking으로 알게 된 스마트폰의 위치 Vector에서 빼주어 손 주변에 스마트폰을 위치시켰습니다.

또한 스마트폰의 회전 상태의 경우 ARcamera에서 modelTarget의 회전 정보를 이용하였는데, Tracking 진행 시에 HMDTrackingScene에서 modelTarget과 ARcamera가 임의로 위치가 바뀌어서 회전하는 축이 VR과 달라 경험적으로 보정을 하였습니다. 그 결과로 y축과 z축이 서로 바뀌었다는 결과를 얻게 되었고 또 x축과 y축이 반대로 움직이는 것을 보아 x와 y에 -1.0f를 곱해주었습니다. HMD tracking을 할 때 HMD의 아랫부분을 찍는 경우가 많아 model부터 아래쪽을 비추는 것을 보고 인식하도록 하여서 x축에 대해 각도가 달라서 경험적으로 더해주어 보정하였습니다.

문제점이 있다면 VR내 스마트폰의 position과 rotiation의 basic은 스마트폰 화면 정중앙이지만, 정작 카메라는 우측 상단에 위치하고 있기 때문에 위치적인 오차도 생기고 스마트폰을 회전만 하더라도 ARcamera 입장에서는 position도 멀어지는 것처럼 보이므로 각도와 위치가 같이 움직입니다. 이렇게 VR화면 상에서와 실제 스마트폰이 정확히 일치하지 않는다는 문제점이 있습니다.

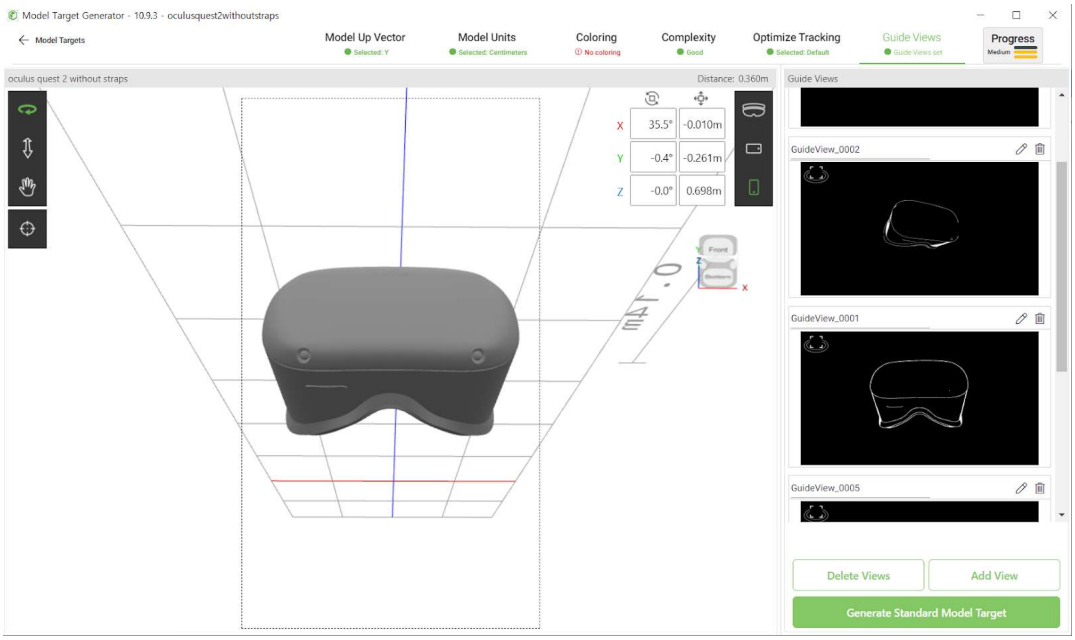
3.2.4. 3D CAD Model



3D CAD Model을 구현할 때 가장 중요하게 생각한 것은 Scaling이였습니다. Model Target Generator(이하 MTG)에서 3D model을 만들 때 실제 크기와 똑같지 않으면 카메라를 통해 물체를 인식할 때 낮은 인식률을 보이기 때문입니다. 만들기 위해 먼저 물체의 실제 크기를 측정했습니다. 끈을 길게하

여 측정했을 때 측정 결과 약 0.19m x 0.10m x 0.29m였고 끈 없이 측정을 하였을 때는 0.19m x 0.10m x 0.14m 였습니다. 두 경우를 다 활용하다 끈이 없을 때 인식을 더 잘하는 경우가 많아 끈을 제거한 HMD 3D CAD Model을 구현했습니다.

3.2.5. Vuforia Model Target Generator



먼저 Autodesk 3dsmax에서 작업한 파일을 MTG에서 작업 가능하도록 파일 확장명을 obj 파일로 변환했습니다. 이후 MTG에 HMD 3D cad model을 model target으로 놓고 실행했습니다. Model target에서 우선 따로 학습을 하지 않아도 물체 인식을 되게끔 하는 guide view mode로 진행하였습니다. 처음에는 정면에서 oculus quest2 를 인식하는 guide view로 작성하여 진행하였으나 여러 방면이 보이는 각도로 guide view를 설정해야 인식률이 높게 나오는 것을 알게 되어 사람들이 주로 핸드폰을 얼굴보다 아래에서 사용하는 것을 생각해 아래에서 위로 바라보며 인식하는 guide view를 설정했습니다. 또한 스마트폰 카메라와 HMD 기기 간 거리도 인식률에 영향을 끼치는 것을 확인하여, guide view에서 거리를 조절하여 완성했습니다.

Advanced guide view는 하나의 물체를 다양한 방면에서 인식을 할 수 있게 끔 하는 모드입니다. 총 4개의 advanced guide view가 존재합니다. 360º Dome, Full 360º, Constrained Angle Range and Target Extent 그리고 Constrained User Positions and Target Extent가 있습니다. 이 중 저희 연구가 사용할 수 있던 방법은 360º Dome과 Constrained Angle Range and

Target Extent 입니다. 360º Dome은 물체의 아래 부분을 제외하면 모든 부분이 Tracking이 가능한 방법입니다. 한정된 각도에서만 Tracking을 하는 것보단 보다 넓은 각도로 Tracking을 하는 것이 더 좋다고 생각하여 진행하였으나 Tracking 정확도가 단일 guide view보다 떨어지는 것을 확인하고 사용하지 않았습니다.

Constrained Angle Range and Target Extent의 경우에는 제한된 각도 범위에서 물체를 탐지하는 것입니다. 이 방법은 기존 guide view에서 보다 상세하게 각도를 설정하여 tracking을 할 수 있으나, 설정하는데 높은 난이도를 보여 기본 guide view로 generate했습니다.

3.2.6. Model Target in Unity

MTG를 통해 만든 3D model을 Vuforia engine package를 설치 후, 기존에 디폴트로 설정되어 있는 Main Camera를 삭제했습니다. 이번 연구에서는 AR 기반으로 tracking을 진행할 것이기 때문에 Main Camera보다는 증강현실을 지원하는 AR Camera가 필요합니다. AR Camera로 교체를 하고 나면, Vuforia Engine을 unity에서 사용할 수 있도록 Vuforia로부터 별도의 라이센스 키를 발급받아야 했습니다. 이후 MTG에서 완성한 model을 import해 AR Camera에 등록했습니다.

Model Target Behaviour script는 3D model을 설정할 수 있는 script입니다. 3D model tracking 성공률을 높이기 위해서는 앞서 언급했듯이 모델의 크기가 실제 HMD 기기 사이즈와 동일해야 했습니다. Script에서 3D model의 크기를 스케일링 할 수 있었지만, 이번 연구에서는 사용하지 않았습니다. Unity 내에서 수치를 조정하게 되면 x, y ,z 3개의 변수가 같이 값이 변동하기에 상세한 수치 조절이 불가능하기 때문입니다. 이러한 상황에 저희는 3D cad model을 외부프로그램에서 별도의 스케일링을 마친 후 사용하였습니다.

테스트를 위해 초기에는 script에서 Model skeleton을 설정하여 tracking이 가능한 각도와 방향을 찾는 작업에 많은 시간을 소요했습니다. 많은 시행착오 끝에 모델의 아랫부분에서 x축으로 약 35º 각도로 스마트폰을 들어 tracking을 하는 것이 가장 높은 인식률을 보였습니다.

그리고 스마트폰 전면 카메라로 tracking을 할 때, 화면 배경이 흰색이면 tracking 성공률이 현저히 떨어졌습니다. Vuforia engine이 3D model과 배경의 경계를 잘 구분하지 못하는 게 이유였고, 추후 진행되는 모든 테스트 는 흰색 배경이 아닌 곳에서 진행했습니다.

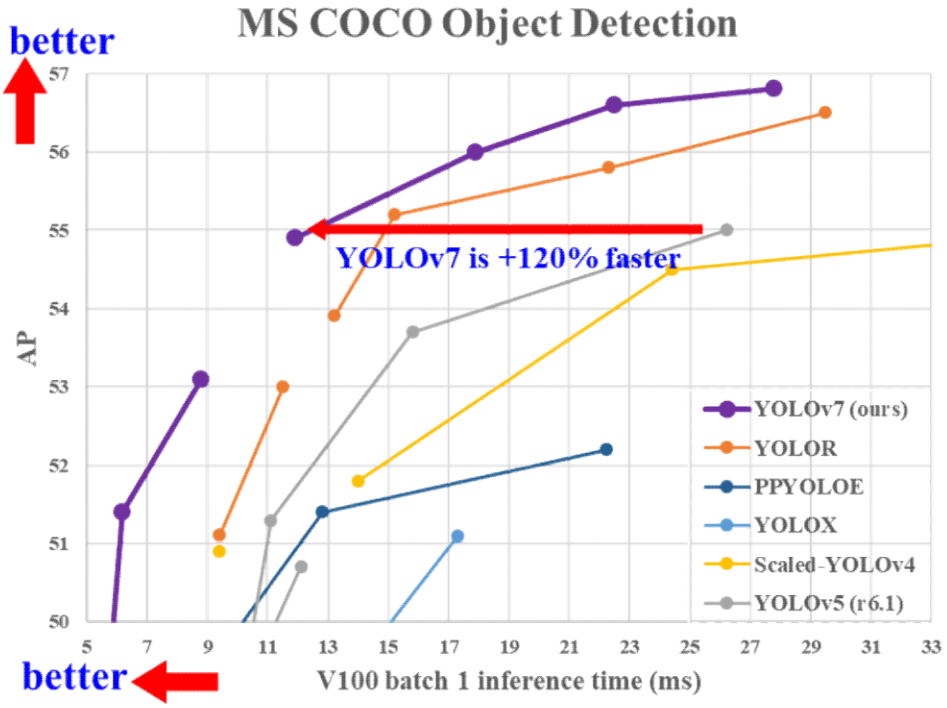
Vuforia engine을 통해 HMD tracking이 된다는 것을 알게 된 후, 스마트폰과 HMD간 거리를 측정할 방법을 생각했습니다. 고려한 방법은 HMD기기 전면부에 납작한 큐브를 부착하여 AR camera와 큐브 간의 좌표 상 거리를계산을 하였습니다.

3.3. Lip motion Tracking

3.3.1. Deep Learning

Lip motion Tracking에는 Deep Learning 을 사용하였습니다. 언어로는 딥러닝 분야에서 자주 쓰이는 Python을 사용하였고, 기술은 YOLOv7을 사용 하였습니다.

해당 Python 프로젝트에선 스마트폰으로부터 이미지 파일을 받아서 입술 검출을 위한 Deep Learning을 거쳐 결과 값을 VR로 전송하는 작업을 합니다. 따라서 Server와 Client, 두 개를 동시에 켜야 하는데 VScode 에서는 같은 폴더에서 동시에 다른 파일을 켜는데 어려움이 있어서 Server와 Client가 다른 수준의 폴더에 존재하도록 하였습니다.



YOLO (You Only Look Once)는 처음으로 one-stage-detection 방법을 고안하여 실시간으로 Object Detection이 가능하게 만들었습니다. 그중에서도

YOLO의 가장 최신 버전인 YOLOv7을 택한 이유는 5fps ~ 160fps 범위에서 YOLO의 이전 버전들을 포함한 지금까지 존재하는 모든 object detector의 성능을 능가하기 때문입니다. YOLOv7은 GPU V100에서 30fps 이상으로 알려진 모든 object detector 중 가장 높은 정확도인 56.8% AP(average precision)를 가지고 있습니다.

저희는 Real-time으로 Lip motion을 최대한 빠르고 정확하게 얻어내길 원했기에, YOLOv7을 선택하였습니다. YOLOv7의 여러 모델 중에서는 가장 기본인 YOLOv7을 사용하였고, YOLOv7은 Pytorch 기반이기에 Pytorch와 Google Colab을 이용해 학습시켰습니다. 학습을 위한 dataset은 Roboflow를 이용하였고, Object detection을 위해 Labeling을 해주었습니다.

Class는 총 세 개로 하였습니다. "closed", "semi-open", "wide-open" 이렇게 세 class로 나눈 dataset을 바탕으로 YOLOv7으로 학습시켰습니다. 처음에는 총 약 100장의 dataset으로 딥러닝을 진행하였는데, 데이터셋이 실사용 가정하에 스마트폰을 손에 들고 찍은 게 아닌 타인이 찍어준 정면 사진들로 구성되어 있었기에 정확도가 떨어졌습니다. 그리하여 학습 dataset을 본 프로젝트에서 저희가 실제로 사용하는 Camera Application을 이용하여 실제 상황을 모방하며 사진 데이터를 재수집하였습니다. 최종적으로 각 class 별로 약 370장의 data가 있는 dataset을 만들었습니다.

딥러닝 안정성을 위해 Unity camera application에서 얻을 수 있는 사진 파일은 1:1 비율로 제한했습니다. batch는 초기에 8, epoch는 32로 하였을 때 신뢰도가 지나치게 낮고 오검출 또한 잦았습니다. 오검출 및 미검출을 줄이기 위해 batch는 16, epoch는 100으로 설정하여 Model을 만들어주었습니다. 가끔 오검출은 되나, 본 프로젝트에서는 미검출이 되지 않는 것이 더 중요하므로 원하는 성능이 나왔다고 판단하여 해당 학습 모델을 최종적으로 선택하였습니다.

Real-time으로 결과를 반환하고 또 Unity에 전달해야 했기에 기존 YOLOv7의 detect.py를 많이 변형했습니다. Unity와 통신하는 코드를 추가하여 가장 신뢰도가 높은 class를 int 형태로 전송하도록 구성하였습니다. 기존의 detect.py는 실행시킬 때 마다 traced_model.pt를 생성하나 저희는 python 환경에서만 model을 load 하기 때문에 traced model이 필요하지 않았기에 해당 코드를 비활성화함으로써 코드의 실행시간을 단축하였습니다.

TCP 통신을 위해 bytearray로 변환해야 했는데, string type의 경우 처리 과

정이 추가되고 값이 제대로 전달되지 않는 문제가 있었습니다. 그래서 처리가 쉬운 int type으로 전달하였습니다. 0이 null 값이므로 기존의 class number에 1씩 더하여 1이 closed, 2이 semi-open, 3이 wide-open 임을 약속하였습니다. 원활한 통신을 위해 기존 통신 코드와 detect.py를 적절히 융합하여 최초 1회 연결 이후 통신이 끊어지지 않은 채로 끊임없이 prediction하여 detection 결과를 얻을 수 있도록 바꾸었습니다.

Unity camera application을 이용, 딥러닝 모델이 있는 python project 상의 사진 파일은 지속적으로 변경됩니다. 그 후 해당 사진 파일에 반복적으로 접근하여 deep learning model에 넣음으로써 lip motion detection 값을 얻는 행위를 반복합니다. 딥러닝 model을 통해 검출된 class들 중 highest confidence를 가진 class index 값을 통신을 통해 전달, unity 캐릭터의 입모양을 구현할 수 있도록 하였습니다.

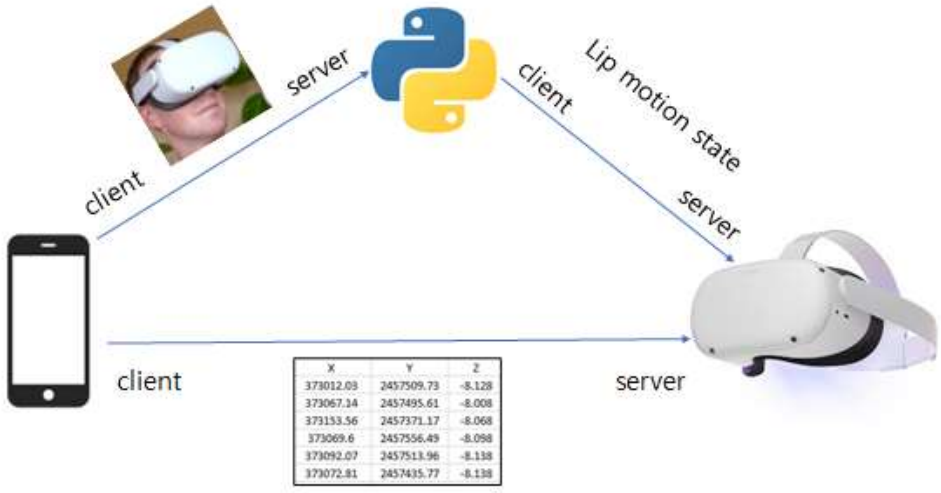
3.3.2. 캐릭터 입모양 제어

캐릭터의 입모양을 움직이기 위해선 당연하겠지만 입모양을 제어할 수 있는 부분이 모델 안에 존재하여야 합니다. 그리고 그 부분은 mouth에 해당하는 object에 Skinned Mesh Renderer의 BlendShapes의 원소들의 값을 변화 시키면 입 모양을 변경시킬 수 있는데, 현 프로젝트에서 사용하는 모델의 경우 Head에 입술 관련 Skinned mesh Renderer가 존재하며 눈썹 위치, 눈 위치, 표정 변화 등 역시 제어할 수 있습니다. 상황에 따라서 입 모양을 바꿔야 하기 때문에 코드로 해당 수치에 접근해야하고 GetBlendShapeWeight, SetBlendShapeWeight 함수가 이를 구현할 수 있게 해줍니다. 입 모양을 바꾸는 BlendShape의 원소는 여러 개가 있지만 그 중 A(발음 "아")와 M(발음 "음")만을 사용하여 구현하였습니다.

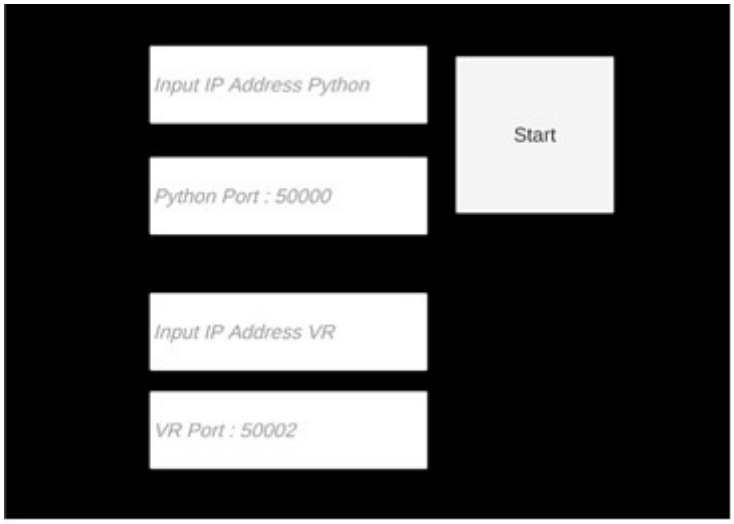
closedMouth를 A를 0으로, M을 100으로 SetBlendShapeWeight함수로 설정하여 구현하고 같은 원리로 wideOpenMouth는 A를 100으로 M을 0으로 하고, semiOpenMouth는 A를 50, M을 100으로 구현하였습니다. 한 순간에 바뀌면 부자연스러우므로 Mathf.Lerp함수를 사용하여 시간에 따라 자연스럽게 선형적으로 증가하도록 구현하였습니다.

4. 연구 결과 분석 및 평가

4.1. Hardware Setting

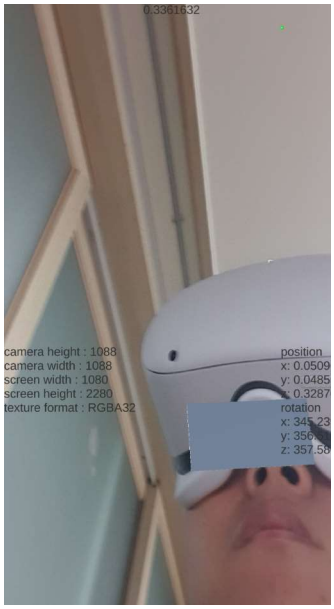


본 연구에서는 TCP 통신을 활용합니다. TCP 통신은 같은 wifi 안에서만 통신이 가능하므로 본 연구에서는 스마트폰, python을 구동하는 컴퓨터, VR이 모두 같은 wifi에 연결되어 있어야 합니다. 그리고 wifi 환경이 바뀔 때마다 python에서 ip주소를 입력하는 코드인 TCP_server.py와 yolov7-main안의 modified_detect.py를 수정해 주어야 합니다.



스마트폰 어플리케이션을 켜서, 첫 번째 빈 칸에는 Python Server의 ip주소를 입력해야 합니다. 두 번째 빈칸에는 python과 통신할 때 port 번호를 입력해야 합니다.세 번째 빈칸에는 VR Server의 ip주소를 입력해야 하고, 네 번째 빈칸에는 VR과 통신할 때의 port 번호를 입력해야 합니다. 모두 입력한 뒤 start 버튼을 누르면 python과 VR이 모두 Server와 연결이 되었을 때 다음 화면으로 넘어갑니다.

4.2. HMD Tracking



HMDTrackingScene에서 Tracking한 HMD앞에 넓은 흰 사각형이 뜨도록 설정하였습니다. 스마트폰과 HMD 사이의 거리는 unity안에서 ARcamera와 사각형의 거리로 얻게 됩니다. 스마트폰 화면의 좌측 중단에는 카메라의 해상도, 화면의 너비와 높이, texture format과 해당 사진 파일의 byte 크기를 표시하였고, 중앙 상단에는 거리를, 우측 중단에는 tracking중인 modelTarget의 position과 rotation Vector의 x,y,z 값을 표시합니다. HMD 중앙에 흰 사각형이 딱 붙어있지는 않더라도 HMD를 따라서 사각형이 움직이는 것을 보아 잘 Tracking하고 있음을 알 수

있습니다.

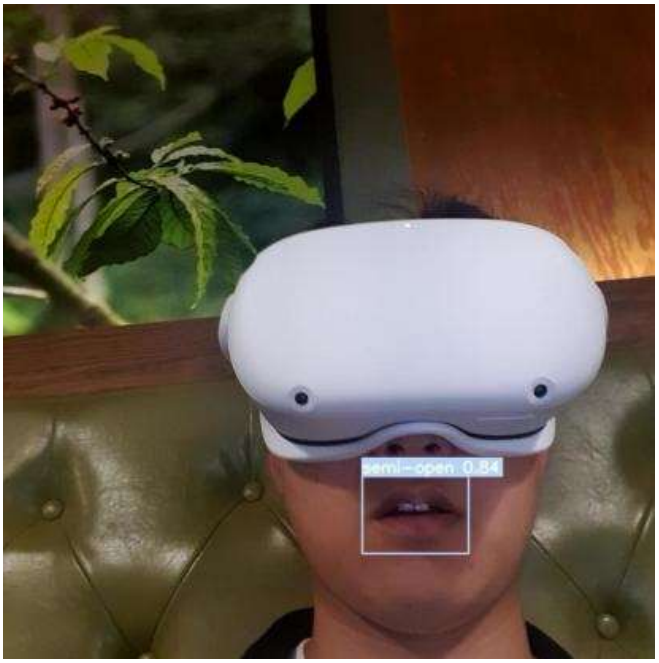
스마트폰 앱으로부터 얻은 HMD의 Vector값을 이용해 3.2.2에서 설명하였듯이 VR 안에서의 스마트폰의 위치를 알 수 있습니다. 그리고 3.3.3에서 말했듯이 보정 과정을 거쳐서 실제 스마트폰 위치에 VR안의 스마트폰이 위치할 수 있도록 조정하였습니다. VR 화면 안에는 여러 Text UI가 있는데 그 중 좌측 하단에 현재 VR안에 있는 스마트폰의 position과 rotation Vector의 x,y,z 값을 표시하였습니다. 이 값이 변하지 않고 가만히 있으면 HMD tracking이 끊긴 것이므로 스마트폰 화면을 다시 조절하여 HMD를 잘 비추도록 위치하면 다시 tracking이 되고 있음을 알 수 있습니다.

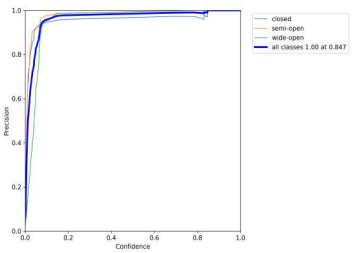
위 사진과 같이 스마트폰을 왼손에 쥐고 검지를 편 상태로 스마트폰을 들고 실험을 진행하였습니다. 우선 왼손에 쥐는 이유는 실험에 사용하는 스마트폰인 갤럭시 s10의 경우 전면 카메라가 우측 상단에 위치하기 때문에 오른손으로 잡았을 경우에 HMD가 전부 찍히지 않을 가능성이 높습니다. 따라서 왼손으로 쥐도록 하였고 검지를 펴는 이유는 HMD가 Hand Tracking을 할때 손가락 정보를 많이 이

용하는데 스마트폰에 가려서 왼손이 VR화면에 보이지 않는 경우가 더러 있습니다. 따라서 왼손의 위치를 잘 보이게 하기 위해 검지를 펴서 실험하였습니다.

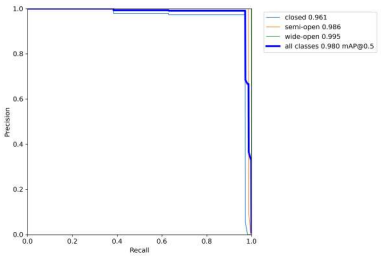
4.3. Lip Motion Tracking

스마트폰의 위치나 각도가 변화해도 wide-open인지 semi-open인지 closed인지 잘 검출해냈습니다. 올바른 값이 들어올 때까지 반복문을 돌도록 처리해두었기에 때문에 간혹 예상치 못하게 발생하는 에러로 인해 종료하는 일이 없었습니다. 다만 당연하게도 입술이 보이지 않으면 검출을 하지 못하므로, 입술이 보일 수 있도록 사용자가 주의하여 스마트폰을 조정해야 합니다. 간혹 전혀 다른 곳을 입술로 인식할 때가 있었으나, 그 경우는 낮은 신뢰도 였기에 결과적으로는 무시되어 문제가 없었습니다. 실제 입술에 대해서는 높은 신뢰도와 정확도를 보였으며, 자세한 수치는 아래 그래프를 통해 확인해 볼 수 있습니다.

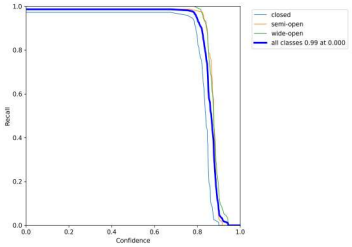




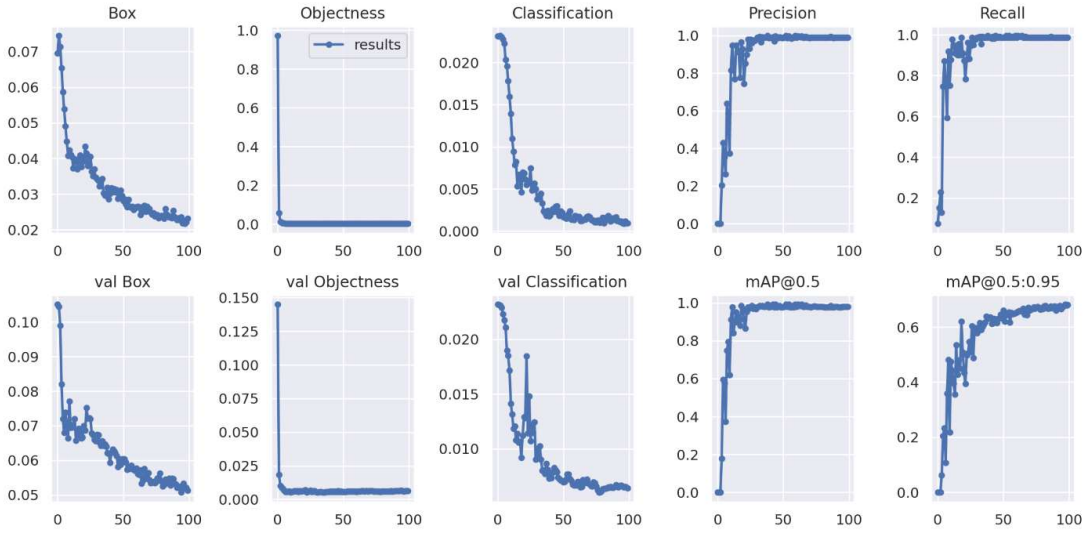
P-curve



PR-curve



R-curve



- closed

VR 화면

실제 입모양 사진

- semi-open

VR 화면

실제 입모양 사진

- wide-open

VR 화면

실제 입모양 사진

5. 결론 및 향후 연구 방향

5.1 결론

저희는 스마트폰 카메라를 활용한 VR 사용자의 가상 현실에서의 몰입도 향상 방안을 연구했습니다.

저희는 전년도 졸업과제에서 향후 연구 과제로 남겨두었던, 스마트폰의 상대적인 위치를 정확히 추정하지 못했다는 점을 이번 연구에서 개선하였습니다. 위 스마트폰의 전면 카메라를 통해 HMD Tracking을 함으로써, HMD를 착용한 사용자가 HMD를 벗지 않고 가상 현실에 있으면서 스마트폰을 이용할 수 있는 기술을 구현했습니다.

Lip motion Tracking 연구를 통해, HMD를 낀 상태에서 스마트폰 카메라로도 Lip motion이 꽤나 정확하게 인식되는 기술을 구현했습니다. 그리고 이를 Tcp 통신과 Unity를 이용해 가상 캐릭터로 구현하여 사용자의 가상현실 몰입감을 더욱 향상시켰습니다.

물론 향후 연구 과제가 남아있으나, 본 연구과제에서 스마트폰 카메라의 활용성을 확인함으로써 스마트폰이 VR / AR 환경에서 몰입도 향상에 기여할 수 있는 잠재력을 확인할 수 있었습니다.

5.2 향후 연구 방향

스마트폰과의 상호작용을 통해 HMD 사용자의 가상환경 상호작용 몰입 개선 방법을 연구하는 것은 분명 의미있지만, 몇 가지 결함들을 보였습니다. 이에 저희는 향후 연구과제로 HMD 카메라로 스마트폰을 tracking, background에서 application을 실행, 그리고 다양한 Lip motion tracking을 통한 케릭터 감정표현입니다.

첫 번째 향후 연구 과제는, 스마트폰으로 Oculus quest 2 내부 카메라를 활용하여 스마트폰이 tracking 하는 것이 아닌, HMD가 스마트폰을 tracking을 하는 것입니다. 스마트폰 카메라를 통해 HMD tracking을 할 때, 카메라에서 HMD가 멀어져 보이지 않게 되면 상호작용을 할 수 없습니다. 그렇기에 Oculus quest2의 카메라를 활용하여 스마트폰을 tracking한다면 상호작용을 할 수 있는 범위가 더 커집니다.

두 번째 향후 연구 과제는, 저희 연구는 application을 가동하여 스마트폰 전면 카메라에 접근하여 HMD를 tracking합니다. 그렇기 때문에, application을 종료하면 HMD tracking을 더 이상 진행 할 수 없습니다. 이러한 문제점을 보완하고자 향후 연구 과제에서는 application을 background에서 작동이 되게하여 다른 application을 사용 할 수 있게 만드는 것입니다.

세 번째 향후 연구 과제는, closed, semi-open, wide-open 이외의 입술모양을 Tracking하여 보다 다양한 상호작용을 할 수 있도록 하는 것입니다.

마지막 향후 연구 과제는, 음성에서 감정을 딥러닝을 통해 분석하여 해당 감정 분석 결과를 통해 가상현실 속 캐릭터가 보다 더 많은 감정표현 나타내도록 하는 것입니다.

6. 참고 문헌

Lowney, M., & Raj, A. S. (n.d.). Model Based Tracking for Augmented Reality on Mobile Devices. Standford.
https://web.stanford.edu/class/ee368/Project_Autumn_1617/Reports/report_lowney_raj.pdf

Wang, C., Bochkovskiy, A., & Liao, H. M. (n.d.). YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. Arxiv.
https://arxiv.org/pdf/2207.02696.pdf