스마트폰을 이용한 HMD 사용자의 가상환경 상호작용 몰입 개선 방법 연구

중간보고서

지도교수 : 이명호

팀명: 띵호와

분과: A

전기컴퓨터공학부 정보컴퓨터공학전공

201924401 강경찬

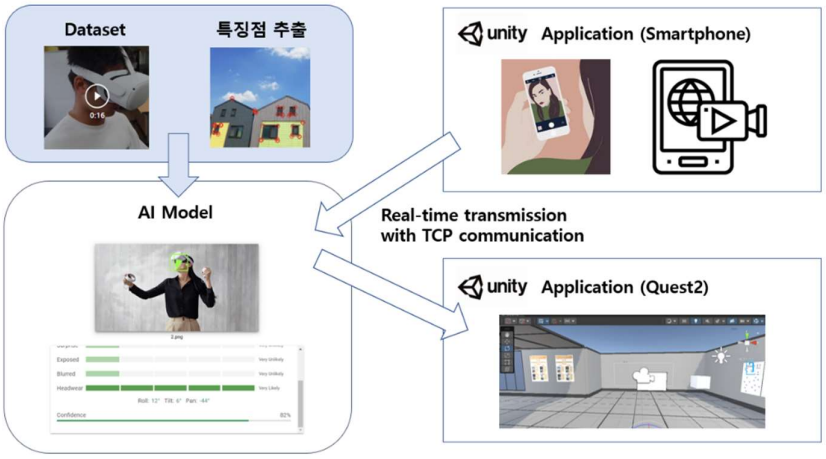
201924472 박예린

201624456 김진서

목 차

1. 요구조건 및 제약 사항 분석에 대한 수정사항

1.1 요구조건 및 그에 대한 수정사항



[그림 1] 과제 전체 구성도

Face Tracking 와 달리 HMD 기기를 착용한 상태의 Face Tracking 연구 결과는 많지 않기에 HMD 기기를 착용한 상태의 얼굴 위치 Tracking 과, 가려지지 않은 부분인 입술의 Motion 을 Tracking 과, 가려지지 않은 부분인 입술의 Motion 을 Tracking 해보려고 한 것이 본 연구의 목표였다. Tracking 을 통해 전 연구과제에서 스마트폰의 위치가 고정되어 있어야 한다는 한계점을 극복하고자 한다.

- HMD 착용 Face Detection and Tracking 과 입술 motion Tracking
  - ■ 두 가지를 병행하려 하였으나, HMD 기기 착용 Face Tracking 의 난이도 문제로 인해 부가 활용 기능 개발용인 Lip motion Tracking 에 대한 연구는 잠정 유보하고 HMD 기기 착용 Face Detection and Tracking 에 중점적으로 시간 투자를 하고자 한다.
  - ■ HMD 기기 착용 Face Detection and Tracking 의 경우 스마트폰의 위치를 보다 정확하게 추정하기 위해서는 HMD 사용자의 얼굴과 스마트폰 간의 거리 뿐만 아니라 그 둘 간의 각도 또한 중요하다. 이미 존재하는 Tracking 라이브러리만 사용해서는 각도 추정이 어렵기에, 특징점을 추출하여 HMD 를 끼고 있는 사용자의 얼굴이 바라보는 각도 측정 CNN 연구가 필요하다.

- Lip / HMD 기기 / HMD 를 끼고 있는 얼굴 중에서 Lip 은 각도를 추정하기에는 어려울 것으로 보이나, 가능하면 세 케이스 모두 특징점을 여러 특징점 추출 이론 기반으로 뽑아보고 각도를 추정하려 한다.

- Unity 를 활용하여 스마트폰 app 구현
  - 스마트폰 카메라를 사용해야하기 때문에 스마트폰 앱을 개발하여야 한다. 그리고 카메라 기능을 접근하여 전면 카메라를 통해 HMD 를 착용한 사용자의 얼굴을 촬영하여야 한다. 그리고 실시간으로 image data 를 server 로 전송해야 한다.

- Unity (Quest2) 상에서 스마트폰을 구현
  - Tracking 으로 얻은 거리 정보와 각도 측정으로 얻은 각도 정보를 바탕으로 Real-time 으로 스마트폰의 위치를 추정하여 구현

- 카메라 접근
  - 추후 추가적인 AR 스마트폰 상호작용 기능 구현을 위해 전/후방 카메라 동시 접근 가능 여부를 Double side Camera 앱을 통하여 가능함을 확인하였다.

1.2 제약 사항 분석에 대한 수정사항

- HMD 기기 착용 Face Detection and Tracking 기능 구현의 어려움
  - 초기 계획: 위 기능과 더불어 Lip motion Tracking 과 추가 기능 개발
  - 변경 계획: 요구 조건에 부합하는 HMD 기기 착용 Face Detection and Tracking 기능 완성이 예상보다 더 난이도가 높기에, 이를 우선적이고 중점적으로 개발하고자 한다.

- 스마트폰 기울어짐에 따른 각도 차이
  - 초기 계획: 스마트폰 기울기 센서를 통해 스마트폰이 얼마나 기울어졌는지에 대한 정보를 반영하여 보완하고자 하였다.

- 변경 계획: 각도를 스마트폰의 전면부 정면이 사용자를 향하게끔, 최대한 기울어지지 않게끔 드는 방향으로 들도록 제한할 것이다.
  - 이유 1) 변수가 줄어들어 스마트폰의 위치를 보다 정확히 추정할 수 있다.
  - 이유 2) 스마트폰을 너무 기울어지게 들면 Tracking 의 기준이 되는 사용자의 얼굴이 카메라에 찍히지 않는 문제가 발생한다.

2. 설계 상세화 및 변경내역

2.1 AI

스마트폰 카메라를 이용한 HMD 기기 tracking 으로 스마트폰과 HMD 기기 상의 거리를 추정하고 이를 통해 스마트폰의 위치를 추정하여 이를 Unity 상에 구현하고자 한다.

이미 존재하는 tracking 라이브러리만 사용해서는 각도 측정이 어려움. HMD 를 끼고 있는 사용자의 얼굴이 바라보는 각도 측정 CNN 이 필요하다. 따라서, 특징점 추출을 통해 ROLL, TILT, PAN 을 측정함으로써 스마트폰의 상대위치를 추정하고자 한다.

2.2 Unity

딥러닝으로 얻어진 HMD 를 착용중인 얼굴 방향(각도)과 스마트폰과의 거리 가지고 VR 안의 가상 스마트폰의 위치와 회전정보를 계산하여 적용할 것이다.

2.3 통신

TCP server 와 client 를 따로 구현하여 VR 프로젝트에서 server 를 구현하고 모바일 앱 프로젝트(Unity)와 딥러닝 프로젝트(Python)에 TCP client 를 구현하여 데이터를 주고받음

3. 갱신된 과제 추진 계획

업무/일정	7 월				8 월				9 월			
	1	2	3	4	1	2	3	4	1	2	3	4
실험용 데이터셋 제작												
HMD tracking												
실제 데이터 수집하기												
direction detection												
Unity (Quest2) 개발												
Unity (스마트폰) 개발												
이미지 전송 최적화												
통신 환경 구현												
테스트 및 디버깅												
최종보고 준비												

4. 구성원별 진척도

이름	역할
박예린	HMD 기기 착용 Face direction Detection Unity - Python 간 통신 데이터 정제 스마트폰 상대 위치 계산
강경찬	HMD VR device 용 Unity 프로그램 개발 촬영 및 데이터 송신용 Unity 안드로이드 앱 개발 TCP 통신 서버구축, client 구현, 데이터 압축 및 전송
김진서	HMD 기기착용 face tracking 라이브러리 활용을 위한 데이터 수집 및 데이터셋 제작 unity client 구현 보조

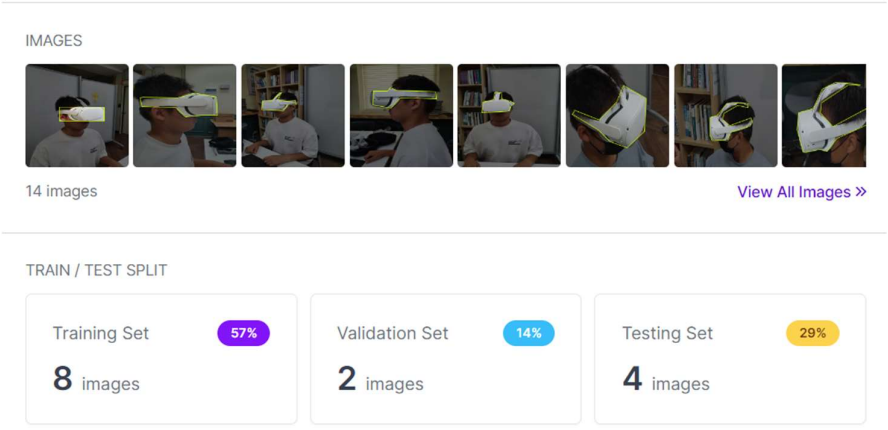
5. 보고 시점까지의 과제 수행 내용 및 중간 결과

5.1 Tracking

5.1.1 YOLOv4 + Deep SORT

YOLO version 4 과 Deep SORT 를 GPU 환경에서 사용하였다. 우선 Pre-trained weights 들을 Tensorflow 모델로 바꾸어, 기존 제공되는 데이터셋과 Class 들을 사용하여 차와 사람을 Tracking 해보았다.

차후 Roboflow 를 통해 커스텀 데이터셋을 만들어, YOLO 를 학습시켜 Tracking 을 해볼 계획이다. Roboflow 는 컴퓨터 비전기술을 활용하여 어플리케이션을 개발할 수 있도록 지원해주는 서비스이며 커스텀 데이터 셋 생성 및 학습에 활용된다.



위 사진은 hmd 기기를 학습시킨 모습이다. 아직은 데이터가 부족하여 낮은 accuracy 를 보인다. 추후에 더 많은 데이터를 수집할 예정이다.

5.1.2 OpenCV Tracker API

YOLOv4 + Deep SORT 를 사용해보기 이전에는 OpenCV 에서 제공하는 Tracker API 들을 사용해보았는데 추적하고자 하는 객체만 지정해주면 API 가 알아서 객체를 추적해주었다.

사용한 API 들은 다음과 같다.

cv2.TrackerBoosting_create(): AdaBoost 알고리즘 기반

cv2.TrackerMIL_create(): MIL(Multiple Instance Learning) 알고리즘 기반

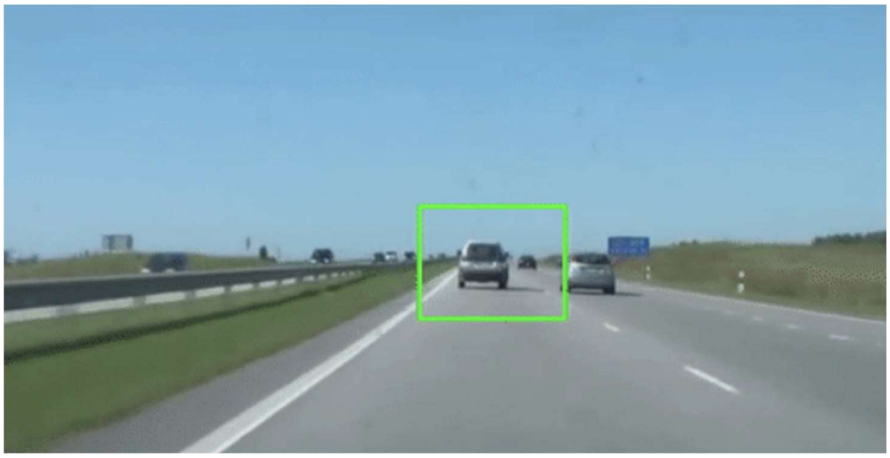
cv2.TrackerKCF_create(): KCF(Kernelized Correlation Filters) 알고리즘 기반

cv2.TrackerTLD_create(): TLD(Tracking, Learning and Detection) 알고리즘 기반

cv2.TrackerMedianFlow_create(): 객체의 전방향/역방향을 추적해서 불일치성을 측정

cv2.TrackerCSRT_create(): CSRT(Channel and Spatial Reliability)

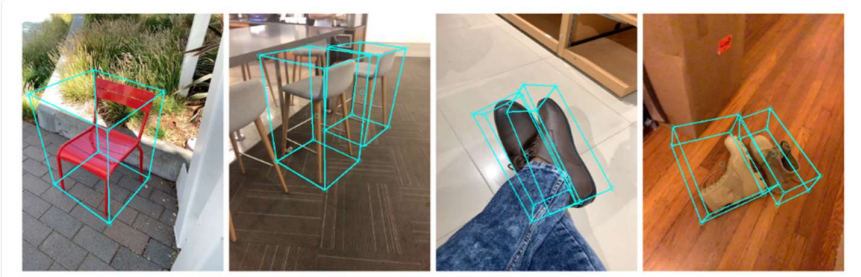
cv2.TrackerMOSSE_create(): 내부적으로 그레이 스케일 사용



그러나 이것들은 YOLO 보다 확연히 정확도가 떨어지는 모습을 보여 사용하지 않을 것 같다.

5.1.3 Mediapipe objectron

Mediapipe objectron 은 hmd 기기를 착용한 상태에서 hmd 기기를 tracking 할 수 있게 도와주는 라이브러리이다. Mediapipe objectron 은 모바일에서 실시간으로 물체 중심의 3D 경계 박스를 나타낸다.



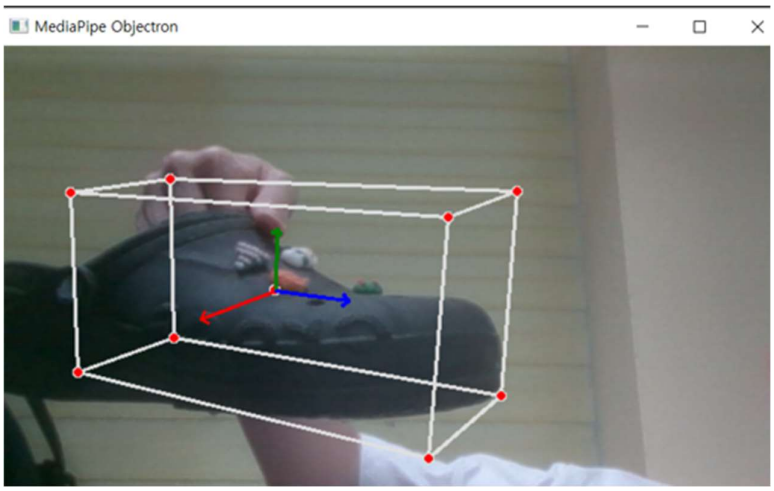
```
# 박스 렌드마크 그리기.
if not results.detected_objects:
  print(f'No box landmarks detected on {file}')
  continue
print(f'Box landmarks of {file}:')
annotated_image = image.copy()
for detected_object in results.detected_objects:
  mp_drawing.draw_landmarks(
      annotated_image, detected_object.landmarks_2d, mp_objectron.BOX_CONNECTIONS)
  mp_drawing.draw_axis(annotated_image, detected_object.rotation,
                       detected_object.translation)
  cv2.imwrite('/tmp/annotated_image' + str(idx) + '.png', annotated_image)
```

```
# Webcam 이용하기
cap = cv2.VideoCapture(0)
with mp_objectron.Objectron(static_image_mode=False,
                            max_num_objects=1,
                            min_detection_confidence=0.5,
                            min_tracking_confidence=0.99,
                            model_name='Shoe') as objectron:
  while cap.isOpened():
    success, image = cap.read()
    if not success:
      print("Ignoring empty camera frame.")
      # webcam 대신 동영상을 불러올 경우 continue 대신 break를 사용.
      continue
```

Webcam 은 스마트폰 카메라로 대체할 수 있다. static_image_mode 를 false 로 설정함으로써 정적인 이미지보단 비디오처럼 동적인 이미지에 적합하다. max_num_objects 는 webcam 으로 탐지할 객체의 최대 개수를 의미한다. HMD 기기만 객체로 인지하면 되기에 1 로 설정한다. model_name 은 추후에 hmd dataset 을 통해 학습시켜 진행할 예정이다.

```
    # 이미지 좌우 반전
    cv2.imshow('MediaPipe Objectron', cv2.flip(image, 1))
    if cv2.waitKey(5) & 0xFF == 27:
      break
cap.release()
```

그리고 이미지 좌우 반전을 통해 사용자가 보기 편한 환경을 조성한다.



위 사진은 신발을 mediapipe objectron 으로 감지한 모습이다. 다른 물체들도 혹시 학습시켜 mediapipe objectron 을 통해 감지를 할 수 있을까 싶어 진행을 해보려고 하였으나 현재로선 mediapipe objectron 으론 구글 데이터셋에서 학습된 물체들만 인식을 한다. 따라서 objectron 은 앞으로 과제에 사용하지 않을 것 같다.

5.2 Face detection

YOLOv4 + DeepSORT 과 같은 Tracking 라이브러리만으로는 얼굴 방향 각도 정보를 얻기 어렵다고 판단했고, Face detection 의 경우에 얼굴 각도를 측정하는 연구 및 API 가 이미 있는 걸 확인했다. 따라서 앞으로 HMD 를 끼고 있는 Face direction 을 de0tection 하기 위해 방향 및 아이디어를 얻고자 수행했다.

5.2.1 Mediapipe Face Detection

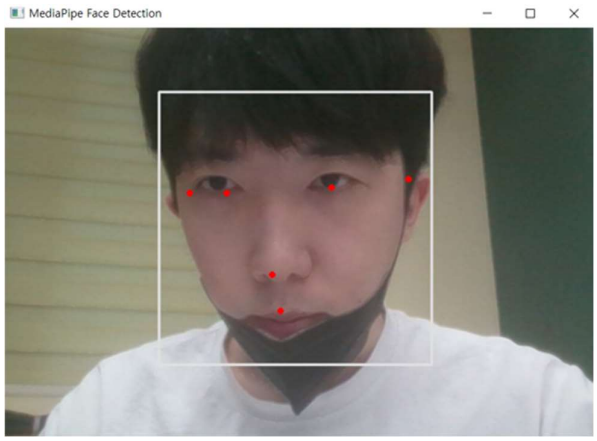
Mediapipe Face detection 은 눈, 코, 입 그리고 귀의 특징점들로 빠르게 얼굴을 인식하는 솔루션이다. 한 번에 여러사람의 얼굴을 인식할 수 있다.

```
with mp_face_detection.FaceDetection(
    model_selection=1, min_detection_confidence=0.5) as face_detection:
```

model_selection 인덱스는 0 또는 1 이다. 0 을 선택하면 2 미터 이내의 근접 촬영에 유용하고 1 을 선택하면 5 미터까지 탐지가 가능하여 전신 촬영에 활용될 수 있다.

```
cap = cv2.VideoCapture(0)
with mp_face_detection.FaceDetection(
    model_selection=0, min_detection_confidence=0.5) as face_detection:
  while cap.isOpened():
    success, image = cap.read()
    if not success:
      print("Ignoring empty camera frame.")
      # If loading a video, use 'break' instead of 'continue'.
      continue
```

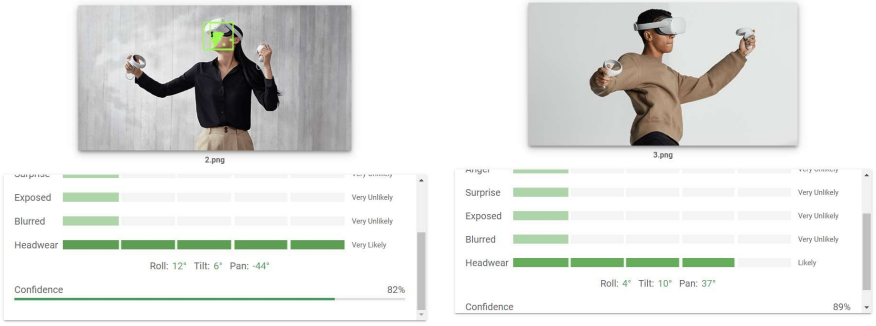
cv2.VideoCapture(0)을 활용해 webcam 을 사용하여 이후에 사용될 스마트폰 카메라를 대체한다.



위 사진은 Mediapipe face detection 을 실행한 모습이다. Quest2 를 쓰고도 mediapipe face detection 을 시도해보았는데, 아래를 바라보는 각도에서는 거의 되지 않았다. 주로 정면을 바라보았을 때는 됐으나 정확도가 높지 않았다. 또한 얼굴 방향 각도 정보도 제공해 주지 않았다.

5.2.2 Google Cloud Vision AI

HMD 기기 중 Quest2 를 끼고 있는 사람들의 다양한 각도의 얼굴사진을 수집, 구글 클라우드에서 제공하는 Vision AI 를 사용해 기기를 끼고 있더라도 Face detection 이 되지 않을까 확인 차 검증을 해보았다.



얼굴이 위를 향하는 각도에서는 얼굴 면적이 Quest2 에 많이 가리지 않아 검출이 되는 반면, 정면을 보고 있으면 현재 얼굴이 바라보는 방향을 정확히 잡지 못하는 경우가 많았다. 이는 Face detection 에 사용되는 특징점(landmark)이 Quest2 에 많이 가려져 있기 때문이라고 판단, Quest2 를 끼고 있는 사진에서 특징점을 새로 잡으려고 한다.

그래서 OpenCV 를 이용하여 해리스 검출을 시도해 보았다.

```
# 해리스 코너 검출 (corner_harris.py)

import cv2
import numpy as np
from google.colab.patches import cv2_imshow
```

```
img = cv2.imread('img/1.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 해리스 코너 검출 ---①
corner = cv2.cornerHarris(gray, 2, 3, 0.04)
# 변화량 결과의 최대값 10% 이상의 좌표 구하기 ---②
coord = np.where(corner > 0.1* corner.max())
coord = np.stack((coord[1], coord[0]), axis=-1)
```

```
cv2_imshow(img)
```

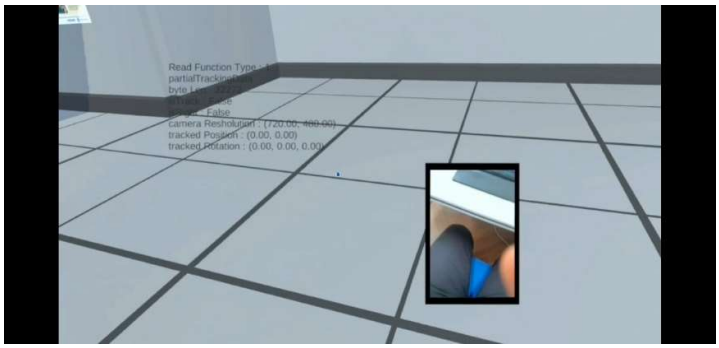
```python
# 시와 토마시 코너 검출 (corner_goodFeature.py)

import cv2
import numpy as np

img = cv2.imread('img/1.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# 시-토마스의 코너 검출 메서드
corners = cv2.goodFeaturesToTrack(gray, 80, 0.01, 10)
# 실수 좌표를 정수 좌표로 변환
corners = np.int32(corners)

# 좌표에 동그라미 표시
for corner in corners:
    x, y = corner[0]
    cv2.circle(img, (x, y), 5, (0,0,255), 1, cv2.LINE_AA)

cv2_imshow(img)
cv2.waitKey()
cv2.destroyAllWindows()
```

위에서 사용하였던 해리스 검출, 시와 토마시 검출 이외에도 OpenCV 에서 제공하는 다양한 검출 방법을 시도해보며, HMD 착용 Face Direction Detection 을 위해 유의미한 특징점만을 뽑아내고, 각각을 비교해보며 성능이 좋은 것을 택하는 방향으로 진행할 것이다.

5.3 Unity

5.3.1 Oculus quest 용 application

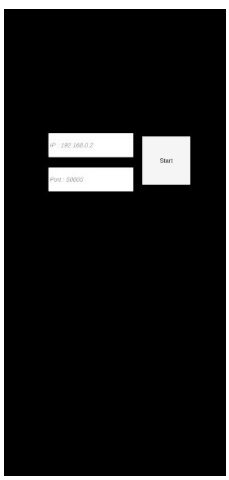
전반적으로 전년도 졸업과제의 뒤를 잇는 과제를 수행중이므로 전년도 졸업과제물을 분석하고 학습하는데 전념하였다.



이는 전년도 졸업과제 프로젝트의 vr 화면이다. 좌측 상단에 떠있는 Text 도 byte len 과 camera Resolution 을 제외한 나머지는 다 삭제하여야 한다. 추후 이미지 파일은 딥러닝 프로젝트에 전송하기위해 최적화를 해야 하므로 해상도나 압축 방식을 다시 고려해야 할 수도 있다.

또한 배경을 다음과 같이 바꾸기로 하였다. 이는 unity asset store 에서 무료로 배포하고 있는 주변환경 에셋이다.

5.3.2 smartphone 용 application



이는 전년도 졸업과제때 사용한 스마트폰 앱 스크린샷이다. 좌측에는 오큘러스용 application 내부에 적힌 ip 를 입력하고 고정 port 인 50000 을 입력하여 start 버튼을 누르면 오른쪽과 같이 카메라에 접근하여 후방 카메라에 비치는 화면을 띄운 것이다. 우리 과제는 스마트폰 카메라의 후방 카메라가 아닌 전방카메라를 사용하므로

앱 프로젝트에 AndroidCamera.cs 에서 SetCamera()를 고쳐야 한다.

```
void SetCamera()
{
    cameraTexture = null;

    WebCamDevice[] devices = WebCamTexture.devices;
    if (devices.Length == 0)
    {
        debugText.text = "There is no available Web Cam";
        Debug.LogError("There is no available Web Cam");
        return;
    }

    for (int i = 0; i < devices.Length; i++)
    {
        if (devices[i].isFrontFacing == isUseFrontCamera)
        {
            cameraTexture = new WebCamTexture(devices[i].name, Screen.width, Screen.height);

            debugText.text = devices[i].availableResolutions.ToString();
            foreach (var resolution in devices[i].availableResolutions)
            {
                Debug.Log("Resolution : " + resolution.ToString());
            }
            break;
        }
    }

    if (cameraTexture == null)
    {
        debugText.text = "There is no available Web Cam";
        Debug.LogError("There is no available Web Cam");
        return;
    }
}
```

5.4 통신

통신 역시 전년도 졸업과제 때 사용했던 것을 분석하여 발전시킬 것이다.

5.4.1 TCP server

- Variables

```
public class TCP_SERVER : MonoBehaviour
{
    private static TCP_SERVER instance = null;

    TcpListener serverSocket;
    TcpClient clientSocket;
    NetworkStream clientStream;
    [SerializeField] int counter = 0;
    [SerializeField] int port;
    [SerializeField] Text debugText;
    [SerializeField] Text debugText2;
    [SerializeField] ModeManager modeManager;
    bool isConnected = false;
    bool isDataReaded = false;
    string readData;
    float fpsCounter = 1;
    int prevCounter = 0;
    int currentFPS = 0;
    [SerializeField] ByteToRawTexture byteToRawTexture;
    private Thread acceptThread, readClientThread;

    private int currentFunctionType;
    private bool currentFunctionUpdated;

    public static TCP_SERVER GetInstance() => instance;
```

FunctionType 이나 ModeManager 의 경우 본 과제에 상황에 맞게 삭제 또는 변경할 예정이다.

- awake & start

```
private void Awake()
{
    if (instance == null)
    {
        instance = this;
    }
    else if (instance != this)
    {
        Destroy(this);
    }
}

// Start is called before the first frame update
void Start()
{
    serverSocket = new TcpListener(port);
    clientSocket = default(TcpClient);
    serverSocket.Start();
    Debug.Log("Server Started");
    counter = 0;
    acceptThread = new Thread(AcceptClient);
    acceptThread.Start();
    readClientThread = new Thread(ReadDataFromClient);
    readClientThread.Start();
    IPHostEntry host = Dns.GetHostEntry(Dns.GetHostName());
    foreach (var ip in host.AddressList)
    {
        if(ip.AddressFamily == AddressFamily.InterNetwork)
            if (debugText.text.CompareTo("Client Connected") != 0)
                debugText.text = "wating" + "\n\nIP is : " + ip.ToString();
    }

    currentFunctionType = 0;
    currentFunctionUpdated = false;
}
```

- update

```
void Update()
{
    if (clientSocket == null)
        return;

    if (!clientSocket.Connected)
    {
        /*
        debugText2.text = "Client Disconnected";
        IPHostEntry host = Dns.GetHostEntry(Dns.GetHostName());
        foreach (var ip in host.AddressList)
        {
            if(ip.AddressFamily == AddressFamily.InterNetwork)
                if (debugText.text.CompareTo("Client Connected") != 0)
                    debugText.text = "wating" + "\nIP is : " + ip.ToString();
        }
        acceptThread.Start();
        */
        return;
    }

    if (currentFunctionType != modeManager.GetCurrentModeMangerFunctionType())
    {
        modeManager.ModeChange(currentFunctionType);
    }

    debugText.text = "Read Function Type : " + currentFunctionType.ToString();
    switch (currentFunctionType)
    {
        case 1:
            debugText.text += "\npartialTrackingData"
                            + "\nbyte Len : " + DataStructs.partialTrackingData.bytesLen.ToString()
                            + "\nisTrack : " + DataStructs.partialTrackingData.isTrack.ToString()
                            + "\nisRight : " + DataStructs.partialTrackingData.isRight.ToString()
                            + "\ncamera Resolution : " + DataStructs.partialTrackingData.cameraResolution.ToString()
                            + "\ntracked Position : " + DataStructs.partialTrackingData.trackedPosition.ToString()
                            + "\ntracked Rotation : " + DataStructs.partialTrackingData.trackedRotation.ToString();
            if (currentFunctionUpdated)
```

```
        {
            byteToRawTexture.LoadPNG(DataStructs.partialTrackingImageData);
            currentFunctionUpdated = false;
        }
        break;
    case 2:
        debugText.text += "\nmirroring Data"
                        + "\nbyte Len : " + DataStructs.mirroringData.bytesLen.ToString()
                        + "\nscreen width : " + DataStructs.mirroringData.screenWidth.ToString()
                        + "\nscreen height : " + DataStructs.mirroringData.screenHeight.ToString();
        if (currentFunctionUpdated)
        {
            byteToRawTexture.LoadPNG(DataStructs.mirroringImageData);
            currentFunctionUpdated = false;
        }
        break;
    case 3:
        debugText.text += "\nTrigger Pressed : " + InputSystem.GetButton(ButtonType.Trigger, ControllerType.LeftController).ToString() +
                        "\nGrip Pressed : " + InputSystem.GetButton(ButtonType.Grip, ControllerType.LeftController).ToString() +
                        "\nButton A Pressed : " + InputSystem.GetButton(ButtonType.ButtonA, ControllerType.RightController).ToString() +
                        "\nButton B Pressed : " + InputSystem.GetButton(ButtonType.ButtonB, ControllerType.RightController).ToString() +
                        "\nright Joystick Axis : " + InputSystem.GetAxis2D(ControllerType.RightController).ToString() +
                        "\nleft Joystick Axis : " + InputSystem.GetAxis2D(ControllerType.LeftController).ToString();
        if (currentFunctionUpdated)
        {
            currentFunctionUpdated = false;
        }
        break;
    case 4:
        foreach (DataStructs.VRKeyboardStruct keyBoardInfo in DataStructs.vrKeyboardBuffor)
        {
            debugText.text += "\nTouch ID : " + keyBoardInfo.touchID.ToString() + ",  Touch Position : " + keyBoardInfo.touchPosition.ToString();
        }
        if (currentFunctionUpdated)
        {
            currentFunctionUpdated = false;
        }
        break;
}
```

update 함수의 경우 functionType 에 따라 case 별로 debugText 와 데이터 전송 코드가 다른데 본 과제는 딥러닝 프로젝트로 보낼 image data 와 딥러닝 프로젝트로부터 받는 좌표 값 data 두 가지이기 때문에 이에 맞게 case 를 조절하고 이에 따른 전송방식도 바꿀 예정이다.

5.4.2 TCP client

- enum & varables

```
public enum FunctionTypes{
    PartialViedoSeeThrough = 1, Mirroring = 2, VRController = 3, VRKeyboard = 4
}


public class TCP_Client : MonoBehaviour
{
    [SerializeField] private ModeChanger modeChanger;
    public InputField IPInput, PortInput;
    public string serverIP;
    public int serverPort;

    private bool socketReady;
    TcpClient socket;
    private NetworkStream stream = null;
    private StreamWriter writer;
    private StreamReader reader;

    private byte[] functionTypeBytes;

    private static TCP_Client instance;

    public static TCP_Client GetInstance() => instance;

    public bool isConnected() => socketReady;
```

앞서 기술했듯 FunctionType 은 본 과제에 맞게 삭제 혹은 수정 할 예정이다. 또한 Modechanger 역시 필요에 따라 삭제할 예정이다.

- awake & update

```
private void Awake()
{
    Time.fixedDeltaTime = 1f / 60f;
}

private void Update()
{
    if (stream == null)
        return;

    if (stream.DataAvailable)
    {
        byte[] functionTypeBytes = BitConverter.GetBytes((int) 0);
        stream.Read(functionTypeBytes, 0, functionTypeBytes.Length);
        int currentFunctionType = BitConverter.ToInt32(functionTypeBytes, 0);
        modeChanger.SceneChagne(currentFunctionType);
    }

}
```

- ConnectToServer

```
public void ConnectToServer()
{
    // 이미 연결되었다면 함수 무시
    if (socketReady) return;

    string ip;
    int port;
    // 기본 호스트 / 포트번호
    if (IPInput != null)
    {
        ip = IPInput.text == "" ? serverIP : IPInput.text;
    }
    else
    {
        ip = serverIP;
    }

    if (PortInput != null)
    {
        port = PortInput.text == "" ? serverPort : int.Parse(PortInput.text);
    }
    else
    {
        port = serverPort;
    }

    // 소켓 생성
    try
    {
        socket = new TcpClient(ip, port);
        socket.SendBufferSize = 12000000;
        stream = socket.GetStream();
        socketReady = true;

    }
    catch (Exception e)
    {
        Debug.Log("Socket Error : " + e.Message);
    }
    Debug.Log("Connected to Server");
    modeChanger.SceneChagne(SceneType.partialVideoSeeThrough);
}
```

- Send

본 과제는 mirroring, vrcontroller, vrkeyboard 를 사용하지 않기 때문에 partialvideoseethrough 부분만 사용하고 다듬을 예정이다.

```
public void Send(FunctionTypes functionType, byte[] bytes = null)
{
    if (!socketReady) return;
    if (!stream.CanWrite) return;

    functionTypeBytes = BitConverter.GetBytes((int)functionType);
    Array.Reverse(functionTypeBytes);

    byte[] dataBytes;

    switch (functionType)
    {
        case FunctionTypes.PartialViedoSeeThrough:
            if (bytes == null)
                return;
            DataStructs.partialTrackingData.bytesLen = bytes.Length;
            Debug.Log("Sending Byte Length : " + bytes.Length);
            dataBytes = DataStructs.StructToBytes(DataStructs.partialTrackingData);

            stream.WriteAsync(functionTypeBytes, 0, functionTypeBytes.Length);
            stream.WriteAsync(dataBytes, 0, dataBytes.Length);
            stream.WriteAsync(bytes, 0, bytes.Length);
            stream.FlushAsync();
            break;
        case FunctionTypes.Mirroring:
            break;

        case FunctionTypes.VRController:
            dataBytes = DataStructs.StructToBytes(DataStructs.vrControllersData);
            stream.WriteAsync(functionTypeBytes, 0, functionTypeBytes.Length);
            stream.WriteAsync(dataBytes, 0, dataBytes.Length);
            stream.FlushAsync();
            break;

        case FunctionTypes.VRKeyboard:
            byte[] byteLengths = BitConverter.GetBytes((int)bytes.Length);
            Array.Reverse(byteLengths);
            stream.WriteAsync(byteLengths, 0, byteLengths.Length);
            stream.WriteAsync(bytes, 0, bytes.Length);
            break;
    }
```