

1. 작품 제목

손 재활 리듬 게임 및 손동작 인지 및 판단 시스템

2. 작품 개요

1. 손 재활이 필요한 환자를 위한 손동작 인지 및 판단 시스템을 구현하고자 한다.
2. 유니티를 이용하여 게임을 만들고 환자가 적당한 위치에서 특정 손 재활 동작을 취하도록 유도하고자 한다.

-손 동작을 판단하고 손 위치를 tracking 하는 것을 첫 번째 목표로 한다.

-판단된 손 동작과 손위치를 클라이언트로 전송하여 게임 진행에 반영하도록 하는 것을 두 번째 목표로 한다.

-게임의 형식을 통해 환자가 수시로 혹은 재미를 느끼며 재활 훈련을 할 수 있도록 유도하는 것을 세 번째 목표로 한다.

기술의 차별점은 다음과 같다.

1. 컴퓨터 내장 웹캠, 혹은 휴대용 웹캠, 거치용 웹캠, 휴대폰 가리지 않고, 게임 실행전 특별한 과정 없이 모두 적용 가능하다.
2. 손가락 마디 사이의 각도를 이용하기 때문에 성별, 나이, 손의 굵기 크기 무관하게 정확한 손 동작 인지 및 판단이 가능하고 이를 통해 정확한 손 재활 동작이 가능하다.
3. 주치의 혹은 사용자가 원하는 특정 손 동작 및 다양한 재활 동작들에 대해 개발한 시스템을 통해 새로 학습시켜 게임에 포함할 수 있다.

게임의 형식을 취함으로써 가지는 차별점은 다음과 같다.

1. 장소와 시간에 구애받지 않고 손 재활 치료를 실시할 수 있다.
2. 게임의 형식으로 매우 다양하게 성취감을 환자에게 주어 치료효과를 높일 수 있다.
4. 의사 혹은 사용자의 따라 환자가 취해야할 자세와 환자 상황에 맞는 유지 시간을 환자가 행동하도록 유도할 수 있다.
5. 실패에 대한 우려 없이 자신의 수행도를 객관적으로 보여줘 다양한 상황에 도전할 기회를 줄 수 있다.
6. 환자가 지치지 않고 치료를 받는 데 도움을 줄 수 있다.

3. 작품 설명

3.1. 주요 동작 및 특징

3.1.1 손동작 인지 및 판단 시스템 동작 과정

1) 손 동작 인지 및 판단

미리 제시되어 있는 예시 동작들을 판단하는 인공지능 모델을 학습한 뒤 학습된 모델을 이용하여 사용자가 사전에 정의된 올바른 손 재활 동작과 비교하여 올바른 동작을 하고 있는지 아니면 제시된 동작과 다른 동작을 취하고 있는지 판단한다.

2) 손 위치 판단

카메라 화면 속에 손을 인지하고 손에서 손목 위치를 판단한다.

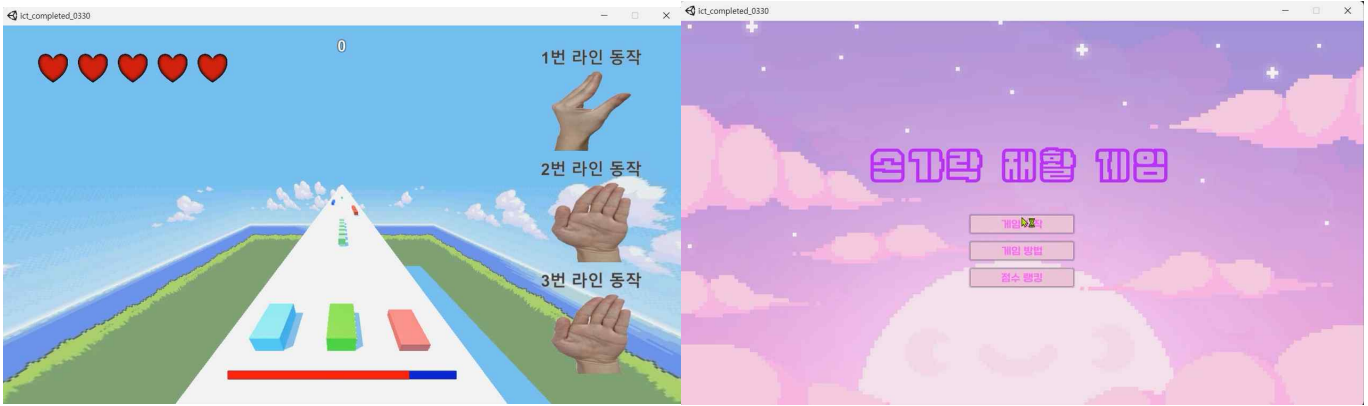
3) TCP 통신을 이용한 손 위치 정보와 손 동작 정보 전달

Python을 서버 Unity를 클라이언트로 정의하여 파이썬에서의 1),2) 결과를 Unity로 전송한다.

4) 전송받은 데이터를 바탕으로 재활 게임 진행

게임에 제시된 동작과 손의 위치를 사용자가 똑같이 따라하게끔 하고 1)2)3) 의 과정을 통해 사용자가 올바른 동작을 취하고 있을 때 포인트를 획득하는 형태의 게임을 진행

3.1.2 게임방법



3.2. 전체 시스템 구성

전체 시스템은 크게 사전 준비하는 동안에 사용되는 시스템 그리고 사용자가 실제로 게임하는 동안에 시스템으로 구분된다.

1. 사전 준비 시스템

사람의 손 동작을 인식하여 손의 마디를 구분하고 각각의 마디를 경유점으로 하는 손 벡터 그리고 각도를 추출하여 전처리 하는 과정

인공지능을 이용하여 모델 학습을 진행하는 과정

2. 게임 진행 시스템

사용자가 취한 동작을 인식하여 데이터를 추출하고 전처리 하여 학습된 모델을 이용하여 판단하는 과정.

판단된 정보를 유니티로 전송하는 과정.

유니티 기반으로 제작된 게임에서 전송받은 데이터를 활용하여 재활 목적으로 게임을 시작하고 진행하는 과정

위와 같은 시스템을 통해 현재 사용자가 게임 화면에 제시되어 있는 손의 동작과 일치하는 동작을 취하고 있는지, 게임에서 제시한 손의 위치가 동일한 위치에서 게임을 진행하고 있는지 판단하여 포인트를 획득하는 방식의 게임을 구현하고 사용자가 조금 더 재미있는 방향으로 재활을 진행할 수 있도록 돕고자 한다.

3.3. 개발환경

3.3.1. 개발 언어

1) Python

Python은 네덜란드에서 개발된 고급 프로그래밍 언어로 독립적인 플랫폼을 사용하는 인터프리터식, 객체지향적 동적 타이핑 대화형 언어이며 가독성이 높고 습득이 빠르다는 장점을 가진 프로그래밍 언어이다. 초보자부터 전문가까지 다양한 사용자를 보유하고 있으며 비슷한 성격인 펄 및 루비와 자주 비교된다. Python의 특징으로는 스크립트 언어를 사용하여 컴파일 과정을 생략이 가능하여 실행 결과를 바로 확인하고 즉각적인 수정이 가능하며 독립적인 플랫폼을 사용하기 때문에 Linux나 Unix, Windows 등 대부분의 OS에서 동작이 가능하다. Python은 사람의 사고방식 및 문법 구사와 유사한 문법을 사용하여 빠르게 학습하고 간결하게 사용이 가능하여 다른 언어에 비해 빠른 개발 속도를 가지고 있으며 C/C++ 등의 다른 개발 언어와 결합하여 사용이 가능해 대표적인 Glue Language에 해당된다. 또한 수많은 표준 라이브러리를 제공하고 다양한 커뮤니티가 형성되어 있어 빠르게 문제를 해결하고 다른 사람들과 협업이 가능하다는 장점을 가지고 있다. 실제 Python은 많은 상용 응용 프로그램에서 스크립트 언어로 채용되고 있으며 Google이나 Instagram, Dropbox 등에서 기본 언어나 다른 언어들과 결합하여 사용 중이다.

1) C#

C#은 Microsoft사에서 개발한 객체 지향 프로그래밍 언어로 Java나 C++과 유사한 점이 많은 프로그래밍 언어이다. C#은 .NET Framework를 이용하여 프로그래밍 하는 대표적인 언어이기도 하며 윈도우 프로그래밍이나 웹, 게임 및 모바일 프로그래밍에서 사용되는 범용 프로그래밍 언어이다. 위에서 설명하였듯이 Java, C++과 기본 문법이 다 언어 스타일 등이 많이 비슷하며 윈도우 뿐만 아니라 안드로이드와 iOS, macOS, Cocoa 최근에는 Linux와 임베디드 분야에서까지 활용이 가능하여 범용성이 매우 높은 언어로 평가받고 있다. 현재는 윈도우 이외에서 C#이 사용가능 하도록 '모노'라는 플랫폼이 개발에 진행중이며 '.NET Core'라는 다른 OS에서도 C#기반 언어가 구동될 수 있도록 도와주는 오픈 소스 프레임워크도 존재한다. 이 프로젝트에서는 유니티 엔진의 기본 개발 언어로 채택되어 사용중이므로 소개하게 되었다.

3.3.2. Tool

1) Unity는 덴마크에서 개발된 2D, 3D 형태의 게임을 개발할 수 있는 환경을 제공하는 게임 엔진이며, 2D, 3D 건축과 애니메이션 나아가 가상현실 등의 미래지향적 콘텐츠 제작을 할 수 있는 통합 제작 엔진이다. 기본 개발 언어로는 C++과 C#을 사용하며 기본 운영 체제로는 Window, macOS, Linux이다. 유니티의 가장 큰 장점은 멀티플랫폼 빌드를 지원한다는 것인데 윈도우 뿐만 아니라 iOS, macOS, Android, Playstation, xbox, Nintendo 등 약 27개의 플랫폼에서 사용이 가능하다는 점이다. 또한 간편하고 간결하게 빌드할 수 있으며 다른 게임 엔진들과는 다르게 비교적 낮은 사양으로 이용이 가능하다. 나아가 사용자 층이 매우 넓고 범용성이 높기 때문에 많은 오픈 소스와 자료들을 찾아볼 수 있으며 다양한 포스트와 강의 영상 등을 찾아볼 수 있다. 마지막으로 저렴한 라이선스 비용으로 사용이 가능하며 초보자들은 무료 버전으로도 충분히 개발이 가능하다. 하지만 현재 보안성과 빈약한 고급성 등이 단점으로 꼽히고 있다.

4. 단계별 제작 과정

4.1 머신러닝을 통한 손 동작 인식(코드 및 설명)

4.1.1 손가락 데이터 추출

```
import cv2,keyboard
import mediapipe as mp
import numpy as np
from tqdm import tqdm

mp_drawing =mp.solutions.drawing_utils
mp_drawing_styles =mp.solutions.drawing_styles
mp_hands =mp.solutions.hands

cap =cv2.VideoCapture(0)

recieve_toggle =False
data_count =0
pbar =tqdm(total=500)

# landmarkdata save
f =open("textfiles/eight.txt",'a')

with mp_hands.Hands(
    model_complexity=0,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5)as hands:

    while cap.isOpened():
        success,image =cap.read()
        if not success:
            print("Ignoring empty camera frame.")
            continue

        # To improve performance, optionally mark the image as not writeable to
        # pass by reference.
        image.flags.writeable =False
        image =cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
        results =hands.process(image)

        # Draw the hand annotations on the image.
        image.flags.writeable =True
        image =cv2.cvtColor(image,cv2.COLOR_RGB2BGR)

        if results.multi_hand_landmarks:

            for hand_landmarks in results.multi_hand_landmarks:
```

```

mp_drawing.draw_landmarks(
    image,
    hand_landmarks,
    mp_hands.HAND_CONNECTIONS,
    mp_drawing_styles.get_default_hand_landmarks_style(),
    mp_drawing_styles.get_default_hand_connections_style())

if recieve_toggle:
    for point in results.multi_hand_landmarks[0].landmark:
        f.write(str(round(point.x,2))+ " "+str(round(point.y,2))+ " "+str(round(point.z,2))+ " ")
        f.write("\n")

    data_count =data_count +1
    pbar.update(1)

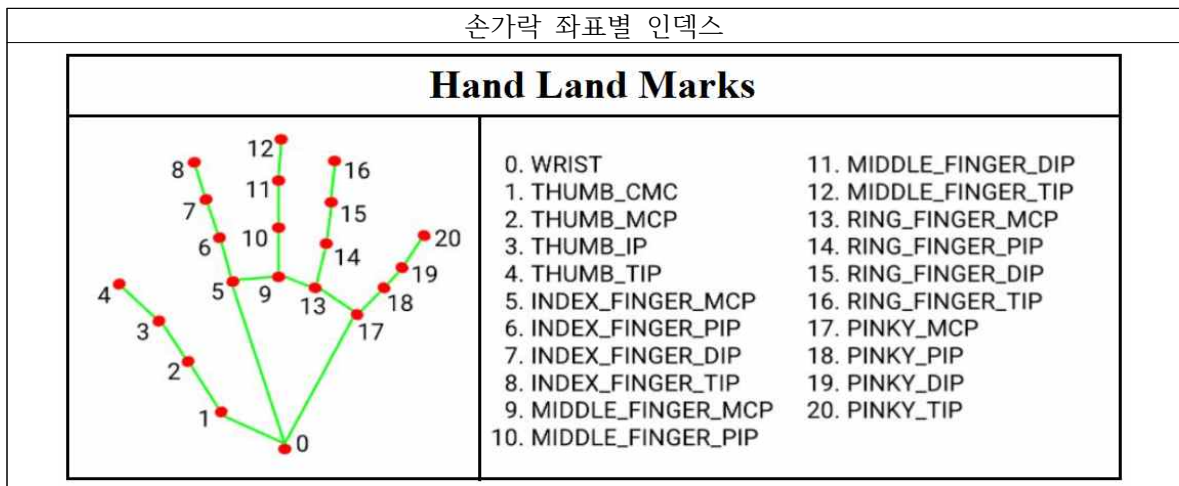
if keyboard.is_pressed("s"):
    recieve_toggle =not recieve_toggle
    print("save start")

# Flip the image horizontally for a selfie-view display.
cv2.imshow('MediaPipe Hands',cv2.flip(image,1))
if cv2.waitKey(1)&0xFF ==27 or data_count ==500:
    print("close")
    f.close()
    pbar.close()
    cv2.destroyAllWindows()
    break

cap.release()

```

media pipe 패키지를 통해 손가락 좌표를 추출 할 수 있는 코드.



버튼 's'버튼을 누를 때마다 다음의 8가지 손동작의 손 마디 좌표를 500개씩 추출한 뒤 각각 텍스트 파일로 저장한다.

* 2번 반복하여 손 마디 좌표를 1000개씩 추출

1. 손가락 벌림	2. 모음굴곡운동	3. 손가락 붙힘	4. 손끝대립운동1
			
5. 손끝대립운동2	6. 손끝대립운동3	7. 손끝대립운동4	8. 총양근운동
			

4.1.2 손가락 데이터 전처리

```
import numpy as np
```

```
one_data =np.array([])
two_data =np.array([])
three_data =np.array([])
four_data =np.array([])
five_data =np.array([])
six_data =np.array([])
seven_data =np.array([])
eight_data =np.array([])
```

```
f =open("textfiles/one.txt",'r')
```

```
while True:
```

```
    line =f.readline()
    f_list =line.split()
    if not line:break
    f_list =np.array(f_list).astype(float)
    one_data =np.append(one_data,f_list)
one_data =np.reshape(one_data,(-1,21,3))
f.close()
```

```
f =open("textfiles/two.txt",'r')
```

```
while True:
```

```
    line =f.readline()
    f_list =line.split()
    if not line:break
    f_list =np.array(f_list).astype(float)
    two_data =np.append(two_data,f_list)
two_data =np.reshape(two_data,(-1,21,3))
f.close()
```

```
f =open("textfiles/three.txt",'r')
```

```
while True:
```

```
    line =f.readline()
    f_list =line.split()
    if not line:break
    f_list =np.array(f_list).astype(float)
    three_data =np.append(three_data,f_list)
three_data =np.reshape(three_data,(-1,21,3))
f.close()
```

```
f =open("textfiles/four.txt",'r')
```

```
while True:
    line =f.readline()
    f_list =line.split()
    if not line:break
    f_list =np.array(f_list).astype(float)
    four_data =np.append(four_data,f_list)
four_data =np.reshape(four_data,(-1,21,3))
f.close()
```

```
f =open("textfiles/five.txt",'r')
```

```
while True:
    line =f.readline()
    f_list =line.split()
    if not line:break
    f_list =np.array(f_list).astype(float)
    five_data =np.append(five_data,f_list)
five_data =np.reshape(five_data,(-1,21,3))
f.close()
```

```
f =open("textfiles/six.txt",'r')
```

```
while True:
    line =f.readline()
    f_list =line.split()
    if not line:break
    f_list =np.array(f_list).astype(float)
    six_data =np.append(six_data,f_list)
six_data =np.reshape(six_data,(-1,21,3))
f.close()
```

```
f =open("textfiles/seven.txt",'r')
```

```
while True:
    line =f.readline()
    f_list =line.split()
    if not line:break
    f_list =np.array(f_list).astype(float)
    seven_data =np.append(seven_data,f_list)
seven_data =np.reshape(seven_data,(-1,21,3))
f.close()
```

```
f =open("textfiles/eight.txt",'r')
```

```
while True:
    line =f.readline()
    f_list =line.split()
    if not line:break
    f_list =np.array(f_list).astype(float)
    eight_data =np.append(eight_data,f_list)
eight_data =np.reshape(eight_data,(-1,21,3))
f.close()
```

위의 4.1.1에서 txt로 저장한 손가락 좌표를 받아 저장

```
import math
```

```
finger_idx ={
    "Thumb":[1,2,3,4],
    "Index_Finger":[5,6,7,8],
    "Middle_Finger":[9,10,11,12],
    "Ring_Finger":[13,14,15,16],
    "Pinky":[17,18,19,20]
```

```

}

def deg_cal(landmark_datas):
    theta =np.array([])

    # z좌표 없애는 기능
    Z_delete =False

    if Z_delete:
        print("z좌표 무시하고 계산")
        k =np.reshape(landmark_datas,(-1,3))
        k =k *(1,1,0)

    landmark_datas =np.reshape(landmark_datas,(-1,21,3))

    for landmark_data in landmark_datas:
        idx =finger_idx["Thumb"]
        v1 =landmark_data[idx[0]]-landmark_data[idx[1]]
        v2 =landmark_data[idx[2]]-landmark_data[idx[1]]
        v3 =landmark_data[idx[1]]-landmark_data[idx[2]]
        v4 =landmark_data[idx[3]]-landmark_data[idx[2]]

        t          h          e          t          a
=np.append(theta,np.rad2deg(np.arccos(np.sum(v1*v2)/(math.dist(v1,(0,0,0))*math.dist(v2,(0,0,0))+1e-10 ))))
        t          h          e          t          a
=np.append(theta,np.rad2deg(np.arccos(np.sum(v3*v4)/(math.dist(v3,(0,0,0))*math.dist(v4,(0,0,0))+1e-10 ))))

        idx =finger_idx["Index_Finger"]
        v1 =landmark_data[idx[0]]-landmark_data[idx[1]]
        v2 =landmark_data[idx[2]]-landmark_data[idx[1]]
        v3 =landmark_data[idx[1]]-landmark_data[idx[2]]
        v4 =landmark_data[idx[3]]-landmark_data[idx[2]]

        t          h          e          t          a
=np.append(theta,np.rad2deg(np.arccos(np.sum(v1*v2)/(math.dist(v1,(0,0,0))*math.dist(v2,(0,0,0))+1e-10 ))))
        t          h          e          t          a
=np.append(theta,np.rad2deg(np.arccos(np.sum(v3*v4)/(math.dist(v3,(0,0,0))*math.dist(v4,(0,0,0))+1e-10 ))))

        idx =finger_idx["Middle_Finger"]
        v1 =landmark_data[idx[0]]-landmark_data[idx[1]]
        v2 =landmark_data[idx[2]]-landmark_data[idx[1]]
        v3 =landmark_data[idx[1]]-landmark_data[idx[2]]
        v4 =landmark_data[idx[3]]-landmark_data[idx[2]]

        #          print(np.rad2deg(np.arccos(np.sum(v2 * v1)/math.dist(v1, (0,0,0))*math.dist(v2, (0,0,0)))))
        t          h          e          t          a
=np.append(theta,np.rad2deg(np.arccos(np.sum(v1*v2)/(math.dist(v1,(0,0,0))*math.dist(v2,(0,0,0))+1e-10 ))))
        t          h          e          t          a
=np.append(theta,np.rad2deg(np.arccos(np.sum(v3*v4)/(math.dist(v3,(0,0,0))*math.dist(v4,(0,0,0))+1e-10 ))))

        idx =finger_idx["Ring_Finger"]
        v1 =landmark_data[idx[0]]-landmark_data[idx[1]]
        v2 =landmark_data[idx[2]]-landmark_data[idx[1]]
        v3 =landmark_data[idx[1]]-landmark_data[idx[2]]
        v4 =landmark_data[idx[3]]-landmark_data[idx[2]]

        t          h          e          t          a
=np.append(theta,np.rad2deg(np.arccos(np.sum(v1*v2)/(math.dist(v1,(0,0,0))*math.dist(v2,(0,0,0))+1e-10 ))))
        t          h          e          t          a
=np.append(theta,np.rad2deg(np.arccos(np.sum(v3*v4)/(math.dist(v3,(0,0,0))*math.dist(v4,(0,0,0))+1e-10 ))))

```

```

idx =finger_idx["Pinky"]
v1 =landmark_data[idx[0]]-landmark_data[idx[1]]
v2 =landmark_data[idx[2]]-landmark_data[idx[1]]
v3 =landmark_data[idx[1]]-landmark_data[idx[2]]
v4 =landmark_data[idx[3]]-landmark_data[idx[2]]

t          h          e          t          a
=np.append(theta,np.rad2deg(np.arccos(np.sum(v1*v2)/(math.dist(v1,(0,0,0))*math.dist(v2,(0,0,0))+1e-10 ))))
t          h          e          t          a
=np.append(theta,np.rad2deg(np.arccos(np.sum(v3*v4)/(math.dist(v3,(0,0,0))*math.dist(v4,(0,0,0))+1e-10 ))))

#개별 손가락 사이의 각도 추가!
v1 =landmark_data[2]-landmark_data[1]
v2 =landmark_data[6]-landmark_data[5]
t          h          e          t          a
=np.append(theta,np.rad2deg(np.arccos(np.sum(v1*v2)/(math.dist(v1,(0,0,0))*math.dist(v2,(0,0,0))+1e-10 ))))

v1 =landmark_data[6]-landmark_data[5]
v2 =landmark_data[10]-landmark_data[9]
t          h          e          t          a
=np.append(theta,np.rad2deg(np.arccos(np.sum(v1*v2)/(math.dist(v1,(0,0,0))*math.dist(v2,(0,0,0))+1e-10 ))))

v1 =landmark_data[10]-landmark_data[9]
v2 =landmark_data[14]-landmark_data[13]
t          h          e          t          a
=np.append(theta,np.rad2deg(np.arccos(np.sum(v1*v2)/(math.dist(v1,(0,0,0))*math.dist(v2,(0,0,0))+1e-10 ))))

v1 =landmark_data[14]-landmark_data[13]
v2 =landmark_data[18]-landmark_data[7]
t          h          e          t          a
=np.append(theta,np.rad2deg(np.arccos(np.sum(v1*v2)/(math.dist(v1,(0,0,0))*math.dist(v2,(0,0,0))+1e-10 ))))

return theta.reshape(len(landmark_datas),-1)

one_data =deg_cal(one_data)
two_data =deg_cal(two_data)
three_data =deg_cal(three_data)
four_data =deg_cal(four_data)
five_data =deg_cal(five_data)
six_data =deg_cal(six_data)
seven_data =deg_cal(seven_data)
eight_data =deg_cal(eight_data)

x_data =np.concatenate((one_data,two_data,three_data,four_data,
                    five_data,six_data,seven_data,eight_data),axis=0)
print(np.shape(x_data))

data_len =1000

y_data =np.concatenate((np.zeros(data_len),np.zeros(data_len)+1,np.zeros(data_len)+2
                    ,np.zeros(data_len)+3,np.zeros(data_len)+4,np.zeros(data_len)+5
                    ,np.zeros(data_len)+6,np.zeros(data_len)+7),axis=0)
print(np.shape(y_data))

import pandas as pd
import numpy as np

d
f
=pd.DataFrame(x_data,columns=['Thumb1','Thumb2','idx1','idx2','mid1','mid2','ring1','ring2','pinky1','pinky2'])

```



```
df['label']=y_data
```

```
df
```

```
df.to_csv("csvfiles/손재활운동2_with_z_position.csv",mode='w')
```

input	문제
손가락 사이의 벡터	벡터의 크기도 영향을 끼치기 때문에 손의 원근에 따라 정확도에 차이가 있음
fig.4-1-1 과 같은 벡터 사이의 각도	class 3번과 8번에서의 각도가 거의 비슷해 두 class간 정확도가 잘 나오지 않음
fig.4-1-2 과 같은 벡터 사이의 각도 4개를 추가	위의 문제 해결

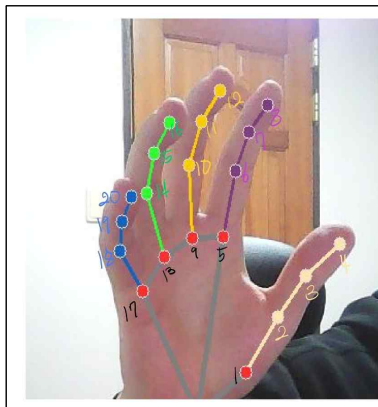


fig.4-1-1

각도 2개
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20
총 10개의 각도

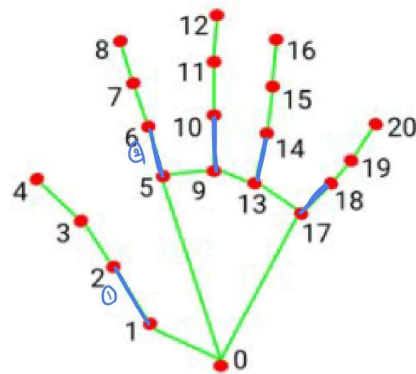


fig.4-1-2

위의 시행착오를 통해 손가락 사이의 총 14개의 각도를 계산하여 1000장씩, 8동작 (8000, 14)형태로 x_data를 만들고 이를 labeling하여 csv 파일로 저장

4.1.3 머신러닝을 통한 다중분류

```
import keras.models
import pandas as pd
import numpy as np
```

```
df =pd.read_csv("csvfiles/손재활운동2_with_z_position.csv")
```

```
df =df.drop('Unnamed: 0',axis=1)
```

```
df.head()
```

```
X =np.array(df[['Thumb1','Thumb2','idx1','idx2','mid1',
                'mid2','ring1','ring2','pinky1','pinky2','angel1','angel2','angel3','angel4']])
```

```
Y =pd.get_dummies(np.array(df['label'])).values
```

```
print("X shape : ",np.shape(X),"y shape : ",np.shape(Y))
```

```
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test =train_test_split(X,Y,
                                                test_size=0.2,
                                                random_state=1)
```

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

```
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.optimizers import Adam
```

```

model = Sequential()

model.add(Dense(64,input_shape=(14,),activation='relu'))
model.add(Dense(64,activation='relu'))
model.add(Dense(8,activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='Adam',
              metrics=['accuracy'])

model.summary()

import matplotlib.pyplot as plt
%matplotlib inline

plt.figure(figsize=(12,8))
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_acc'])
plt.legend(['loss','val_loss','acc','val_acc'])
plt.grid()
plt.show()

loss,accuracy =model.evaluate(X_test,y_test)
print("Accuracy = {:.2f}".format(accuracy))

from keras.models import load_model

model.save('weights/손재활운동.h5')

```

위의 4.1.2에서 저장한 csv파일을 받아 온 뒤 모델에 넣이 적절한 형태로 변환

	Thumb1	Thumb2	idx1	idx2	mid1	mid2	ring1	ring2	pinky1	pinky2	angel1	angel2	angel3	angel4	label
0	175.025569	159.171031	176.486840	171.948376	175.108138	172.519484	167.976219	172.923608	168.578240	164.774835	24.116013	7.749370	15.476119	111.945850	0.0
1	175.025569	159.171031	169.765193	167.976215	178.754607	176.633520	167.976219	167.976215	169.254974	171.793257	24.884200	11.488882	10.842936	111.945850	0.0
2	177.737517	153.594435	170.946349	167.616153	176.784505	179.989360	174.877773	179.988053	160.769324	155.530518	26.534036	10.673981	10.673981	111.945850	0.0
3	177.737517	153.594435	176.486840	174.877764	176.784505	179.989360	174.877773	175.479754	160.769324	155.530518	24.116013	14.217195	10.673981	111.945850	0.0
4	177.737517	153.594435	178.336581	174.877764	173.418049	176.633520	167.976219	162.281385	173.622697	171.793257	25.440719	12.071890	10.673981	112.469779	0.0

csv파일의 head() 데이터

다음의 모델을 통해 epoch = 1000으로 하여 학습 진행

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	960
dense_1 (Dense)	(None, 64)	4160
dense_2 (Dense)	(None, 8)	520

Total params: 5,640
Trainable params: 5,640
Non-trainable params: 0

```
Epoch 1/1000
200/200 [=====] - 1s 4ms/step - loss: 2.5199 - accuracy: 0.7572 - val_loss: 0.2290 - val_accuracy: 0.8831
Epoch 2/1000
200/200 [=====] - 0s 2ms/step - loss: 0.2142 - accuracy: 0.9175 - val_loss: 0.1238 - val_accuracy: 0.9531
Epoch 3/1000
200/200 [=====] - 0s 2ms/step - loss: 0.1527 - accuracy: 0.9398 - val_loss: 0.0736 - val_accuracy: 0.9731
Epoch 4/1000
200/200 [=====] - 0s 2ms/step - loss: 0.1791 - accuracy: 0.9400 - val_loss: 0.2167 - val_accuracy: 0.9106
Epoch 5/1000
200/200 [=====] - 0s 2ms/step - loss: 0.1816 - accuracy: 0.9337 - val_loss: 0.1713 - val_accuracy: 0.9212
Epoch 6/1000
200/200 [=====] - 0s 2ms/step - loss: 0.0943 - accuracy: 0.9856 - val_loss: 0.0450 - val_accuracy: 0.9856
Epoch 7/1000
200/200 [=====] - 0s 2ms/step - loss: 0.0147 - accuracy: 0.9958 - val_loss: 0.0194 - val_accuracy: 0.9919
Epoch 8/1000
200/200 [=====] - 0s 2ms/step - loss: 0.0109 - accuracy: 0.9961 - val_loss: 0.0131 - val_accuracy: 0.9962
Epoch 9/1000
200/200 [=====] - 0s 2ms/step - loss: 0.0132 - accuracy: 0.9967 - val_loss: 0.0148 - val_accuracy: 0.9962
Epoch 10/1000
200/200 [=====] - 0s 2ms/step - loss: 0.0066 - accuracy: 0.9978 - val_loss: 0.0111 - val_accuracy: 0.9962
Epoch 11/1000
200/200 [=====] - 0s 2ms/step - loss: 0.0075 - accuracy: 0.9981 - val_loss: 0.0498 - val_accuracy: 0.9887
Epoch 12/1000
200/200 [=====] - 0s 2ms/step - loss: 0.0076 - accuracy: 0.9970 - val_loss: 0.0194 - val_accuracy: 0.9937
```



test data를 넣어 정확도가 잘 나오는 지 확인 후 모델을 저장

```
50/50 [=====] - 0s 2ms/step - loss: 0.0194 - accuracy: 0.9937
Accuracy = 0.99
```

4.1.4 실시간 손동작 예측 및 유니티와 TCP 통신

```
import cv2, os, re, math
import mediapipe as mp
import numpy as np
```

```
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_hands = mp.solutions.hands
```

```
from keras.models import load_model
model = load_model('weights/손재활운동손가락사이각도포함.h5')
```

```
import socket
import time
from datetime import datetime
```

```
HOST = '127.0.0.1'
PORT = 8000
```

```
print("연결 대기 상태입니다")
```

```
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
```

```
server_socket.bind((HOST, PORT))
```

```
server_socket.listen()
```

```

client_socket, addr = server_socket.accept()

print('Connected by', addr)

finger_idx = {
    "Thumb" : [1,2,3,4],
    "Index_Finger" : [5,6,7,8],
    "Middle_Finger" : [9,10,11,12],
    "Ring_Finger" : [13,14,15,16],
    "Pinky" : [17,18,19,20]
}

def deg_cal(landmark_datas):
    theta = np.array([])

    # z좌표 없애는 기능
    Z_delete = False

    if Z_delete:
        print("z좌표 무시하고 계산")
        k = np.reshape(landmark_datas, (-1,3))
        k = k * (1,1,0)

    landmark_datas = np.reshape(landmark_datas, (-1,21,3))

    for landmark_data in landmark_datas:
        idx = finger_idx["Thumb"]
        v1 = landmark_data[idx[0]] - landmark_data[idx[1]]
        v2 = landmark_data[idx[2]] - landmark_data[idx[1]]
        v3 = landmark_data[idx[1]] - landmark_data[idx[2]]
        v4 = landmark_data[idx[3]] - landmark_data[idx[2]]

        theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )
        theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v3*v4) / ( math.dist(v3,(0,0,0)) *
math.dist(v4,(0,0,0)) + 1e-10 ) ) ) )

        idx = finger_idx["Index_Finger"]
        v1 = landmark_data[idx[0]] - landmark_data[idx[1]]
        v2 = landmark_data[idx[2]] - landmark_data[idx[1]]
        v3 = landmark_data[idx[1]] - landmark_data[idx[2]]
        v4 = landmark_data[idx[3]] - landmark_data[idx[2]]

        theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )
        theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v3*v4) / ( math.dist(v3,(0,0,0)) *
math.dist(v4,(0,0,0)) + 1e-10 ) ) ) )

        idx = finger_idx["Middle_Finger"]
        v1 = landmark_data[idx[0]] - landmark_data[idx[1]]
        v2 = landmark_data[idx[2]] - landmark_data[idx[1]]
        v3 = landmark_data[idx[1]] - landmark_data[idx[2]]
        v4 = landmark_data[idx[3]] - landmark_data[idx[2]]

        # print(np.rad2deg(np.arccos(np.sum(v2 * v1)/math.dist(v1, (0,0,0))*math.dist(v2, (0,0,0)))))
        theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )
        theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v3*v4) / ( math.dist(v3,(0,0,0)) *
math.dist(v4,(0,0,0)) + 1e-10 ) ) ) )

```

```

        idx = finger_idx["Ring_Finger"]
        v1 = landmark_data[idx[0]] - landmark_data[idx[1]]
        v2 = landmark_data[idx[2]] - landmark_data[idx[1]]
        v3 = landmark_data[idx[1]] - landmark_data[idx[2]]
        v4 = landmark_data[idx[3]] - landmark_data[idx[2]]

        theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )
        theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v3*v4) / ( math.dist(v3,(0,0,0)) *
math.dist(v4,(0,0,0)) + 1e-10 ) ) ) )

        idx = finger_idx["Pinky"]
        v1 = landmark_data[idx[0]] - landmark_data[idx[1]]
        v2 = landmark_data[idx[2]] - landmark_data[idx[1]]
        v3 = landmark_data[idx[1]] - landmark_data[idx[2]]
        v4 = landmark_data[idx[3]] - landmark_data[idx[2]]

        theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )
        theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v3*v4) / ( math.dist(v3,(0,0,0)) *
math.dist(v4,(0,0,0)) + 1e-10 ) ) ) )

        #개별 손가락 사이의 각도 추가!
        v1 = landmark_data[2] - landmark_data[1]
        v2 = landmark_data[6] - landmark_data[5]
        theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )

        v1 = landmark_data[6] - landmark_data[5]
        v2 = landmark_data[10] - landmark_data[9]
        theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )

        v1 = landmark_data[10] - landmark_data[9]
        v2 = landmark_data[14] - landmark_data[13]
        theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )

        v1 = landmark_data[14] - landmark_data[13]
        v2 = landmark_data[18] - landmark_data[7]
        theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )

    return theta.reshape(len(landmark_datas), -1)

cap = cv2.VideoCapture(0)
deg = []

idx = 0

with mp_hands.Hands(
    model_complexity=0,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5) as hands:

    while cap.isOpened():
        success, image = cap.read()
        if not success:
            print("Ignoring empty camera frame.")
            continue

```

```

# To improve performance, optionally mark the image as not writeable to
# pass by reference.
image.flags.writeable = False
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
results = hands.process(image)

# Draw the hand annotations on the image.
image.flags.writeable = True
image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

if results.multi_hand_landmarks:

    for hand_landmarks in results.multi_hand_landmarks:
        mp_drawing.draw_landmarks(
            image,
            hand_landmarks,
            mp_hands.HAND_CONNECTIONS,
            mp_drawing_styles.get_default_hand_landmarks_style(),
            mp_drawing_styles.get_default_hand_connections_style())
        # print(hand_landmarks)

    print(results.multi_hand_landmarks[0].landmark[0].x)
    handPosition=int(results.multi_hand_landmarks[0].landmark[0].x*100)
    print(handPosition)
    arr = np.array([])

    for point in results.multi_hand_landmarks[0].landmark:
        arr = np.append(arr, [round(point.x, 2), round(point.y, 2), round(point.z, 2)])
    arr = np.reshape(arr, (-1,3))
    deg = deg_cal(arr)
    y_prob = model.predict(deg)
    predicted = y_prob.argmax(axis=-1)
    print(predicted[0])
    msg=str(int(predicted[0]) + 1)+str(handPosition)
    # msg = str(int(predicted[0]) + 1)+'.'+str(results.multi_hand_landmarks[0].landmark[0].x)
    print(msg)
    client_socket.sendall(msg.encode())

# Flip the image horizontally for a selfie-view display.
img = cv2.flip(image, 1)

















cv2.imshow('MediaPipe Hands', img)

if cv2.waitKey(1) & 0xFF == 27:
    print("close")
    cv2.destroyAllWindows()
    break

```

cap.release()

파이썬-server/유니티-client로 하여 요청이 오면 실시간으로 예측 데이터 및 현재 손의 위치정보의 데이터를 전송한다.

동작	실시간 예측(왼쪽 위 predict)	
손가락 벌림		
모음굴곡운동		
손가락 붙임		
손끝대립운동1		
손끝대립운동2		
손끝대립운동3		
손끝대립운동4		
충양근운동		

파이썬 -> 유니티 데이터 전송



현재 손가락 벌림운동 중 class 1, 손 위치 약 우측에서부터 53% 위치
출력 확인

4.2 Unity를 통한 game 구현(코드 및 설명)

```
using System.Collections;
using System.Collections.Generic;
using System.Net.Sockets;
using UnityEngine;
using System.Text;
using System;
using System.IO;
using System.Runtime.InteropServices;

public class TCPReceive : MonoBehaviour
{
    //TCP 통신 관련 변수
    TcpClient client;
    string serverIP = "127.0.0.1";
    int port = 8000;
    byte[] receivedBuffer;
    bool socketReady = false;
    NetworkStream stream;

    //User defined
    public int predict = 0;
    public int HandLocation=50;

    // Start is called before the first frame update
    void Start()
    {
        CheckReceive();
    }
}
```



```

}

// Update is called once per frame
void Update()
{
    if(socketReady)
    {
        if (stream.DataAvailable)
        {
            receivedBuffer = new byte[3];
            stream.Read(receivedBuffer, 0, receivedBuffer.Length); // stream에 있던 바이트배열 내
러서 새로 선언한 바이트배열에 넣기
            string msg = Encoding.UTF8.GetString(receivedBuffer, 0, receivedBuffer.Length); //
byte[] to string
            // predict = msg;
            // Debug.Log(msg);
            // Debug.Log(msg.Substring(0,1));
            // Debug.Log(msg.Substring(1));
            predict=int.Parse(msg.Substring(0,1));
            HandLocation=int.Parse(msg.Substring(1));
            // Debug.Log(predict);
        }
    }
}

void CheckReceive()
{
    if (socketReady) return;
    try
    {
        client = new TcpClient(serverIP, port);

        if (client.Connected)
        {
            stream = client.GetStream();
            Debug.Log("Connect Success");
            socketReady = true;
        }
    }
    catch (Exception e)
    {
        Debug.Log("On client connect exception " + e);
    }
}

```

```

void OnApplicationQuit()
{
    CloseSocket();
}

void CloseSocket()
{
    if (!socketReady) return;

    client.Close();
    socketReady = false;
}
}

```

본 게임에서는 우선 8개의 동작을 사전 트레이닝 하여 진행하였다. 서버(파이썬)에서 보낸 데이터는 다음과 같은 형태를 가진다.

사용자가 취한 동작의 label	사용자 순위 x축 위치
-------------------	--------------

이 두 개의 정보를 담은 데이터를 파이썬에서 보낸다.

예시를 들자면 다음과 같다.

사용자가 1번 동작을 취하고 화면의 중간(50%) 위치에 손을 둔다면 데이터는 “150” 이라는 문자열 데이터가 된다. 이를 slice 하여 구분하여 게임 진행에 사용한다.

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ButtonController : MonoBehaviour
{
    // public KeyCode keyToPress;
    [SerializeField] Animator button2PressAnimator=null;

    public string press2="Press2";
    public string pull2="Pull2";
    public string NoteTag="Note3D2";

    public static bool canHit2=false;
    public int currentState =0;
    SliderController theSliderController;
    ObjectManager theObjectManager;
    ScoreManager theScoreManager;
    ComboManager theComboManager;
    NoteManager3D theNoteManager3D;
    DesiredHandController theDesiredHandController;
    TCPReceive theTCPReceive;
}

```

```

StatusManager theStatus;
void Start()
{
    theObjectManager=GetComponent<ObjectManager>();
    theSliderController=FindObjectOfType<SliderController>();
    theScoreManager=FindObjectOfType<ScoreManager>();
    theComboManager=FindObjectOfType<ComboManager>();
    theNoteManager3D=FindObjectOfType<NoteManager3D>();
    theTCPReceive=FindObjectOfType<TCPReceive>();
    theDesiredHandController=FindObjectOfType<DesiredHandController>();
    theStatus=FindObjectOfType<StatusManager>();
}

// Update is called once per frame
void Update()
{
    // Debug.Log("theSliderController.value"+theSliderController.theCurrentPosition);

    // if(Input.GetKeyDown(keyToPress))
    // {
    //     // button2PressAnimator();

    //     if(theSliderController.theCurrentPosition>=0.33    &&
theSliderController.theCurrentPosition <= 0.66)
    //     {
    //         button2PressAnimator.SetTrigger(press2);
    //         canHit2=true;
    //     }
    //     else{
    //         canHit2=false;
    //     }

    // }
    // if(Input.GetKeyUp(keyToPress))
    // {
    //     // theSR.sprite=defaultImage;

    //     if(theSliderController.theCurrentPosition>=0.33    &&
theSliderController.theCurrentPosition <= 0.66)
    //     {
    //         button2PressAnimator.SetTrigger(pull2);
    //     }
    //     canHit2=false;
    // }

    if(theSliderController.theCurrentPosition>=0.33 && theSliderController.theCurrentPosition <=
0.66)

```

```

{

    if(theTCPReceive.predict==((int)theDesiredHandController.image2Idx+1))
    {
        // Debug.Log("buttonController2 Hit");

        canHit2=true;
        if(currentState==0)
        {
            button2PressAnimator.SetTrigger(press2);
            currentState=1;
        }
    }
    else{

        canHit2=false;
        if(currentState==1)
        {
            button2PressAnimator.SetTrigger(pull2);
            currentState=0;
        }
    }
}
else{
    canHit2=false;
    if(currentState==1)
    {
        button2PressAnimator.SetTrigger(pull2);
        currentState=0;
    }
}
}

public void Button2PressEffect()
{
    button2PressAnimator.SetTrigger(press2);
}

private void OnTriggerEnter(Collider collision)
{
    // theNoteManager3D.LastNote3DLine=2;
    if(collision.CompareTag(NoteTag))
    {
        int i=collision.gameObject.GetComponent<Note3DController>().lastTest();
        if(i==1)
    }
}

```

```

    {
        theDesiredHandController.hand2OnIng=0;
    }
    if (canHit2)
    {

        // canBePressed=false;
        // Debug.Log("OnTriggerExit3D");
        theScoreManager.IncreaseScore();
        Destroy(collision.gameObject);
        // theObjectManager.boxNoteList.Remove(collision.gameObject);


        // GameManager.instance.NoteMissed();
    }
    else{

    }

}
}
private void OnTriggerExit(Collider collision)
{

    if(collision.CompareTag(NoteTag))
    {
        // canBePressed=false;
        // Debug.Log("OnTriggerExit3D");
        theStatus.DecreaseHp(1);
        theComboManager.ResetCombo();


        // theObjectManager.boxNoteList.Remove(collision.gameObject);


        // GameManager.instance.NoteMissed();
    }

}
}

```

각 라인마다 정답 손동작이 있고 실시간으로 사용자 손이 어느 버튼 위치에 있는지를 표시해준다. 이 때 사용자의 손동작이 올바르게 해당 버튼이 눌리고 눌린 상태에서 위에서 내려오는 박스와 충돌하면 박스를 파괴하면서 점수가 상승한다.

5. 사용한 제품 리스트(제품명, 제품 링크)

	<p>< 제품명 > Logitech HD webcam C270</p> <p>< 제품 구매 링크 > https://www.devicemart.co.kr/goods/view?no=1099029</p>
	<p>< 제품명 > [하나아크릴] 3T 포맥스</p> <p>< 제품 구매 링크 > https://www.devicemart.co.kr/goods/view?no=10114</p>
	<p>< 제품명 > [하나아크릴] 3T 포맥스</p> <p>< 제품 구매 링크 > https://www.devicemart.co.kr/goods/view?no=7940</p>
	<p>< 제품명 > 팔 및 손목 지지대</p> <p>※ 필요에 따라 장기간 손을 들고 있기 불편한 사용자를 위해 팔과 손목을 지지할 수 있는 제품</p>
	<p>< 제품명 > 웹캠 지지대</p> <p>※ 고정된 웹캠을 사용하지 않아 웹캠의 고정 및 지지가 필요할 경우 사용</p>

6. 기타

6.1. 최종 코드(파이썬)

```
import cv2, os, re, math
import mediapipe as mp
import numpy as np

mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_hands = mp.solutions.hands

from keras.models import load_model
model = load_model('weights/손재활운동손가락사이각도포함.h5')

import socket
import time
from datetime import datetime

HOST = '127.0.0.1'
PORT = 8000

print("연결 대기 상태입니다")

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

server_socket.bind((HOST, PORT))

server_socket.listen()

client_socket, addr = server_socket.accept()

print('Connected by', addr)

finger_idx = {
    "Thumb" : [1,2,3,4],
    "Index_Finger" : [5,6,7,8],
    "Middle_Finger" : [9,10,11,12],
    "Ring_Finger" : [13,14,15,16],
    "Pinky" : [17,18,19,20]
}

def deg_cal(landmark_datas):
    theta = np.array([])

    # z좌표 없애는 기능
```

```

Z_delete = False

if Z_delete:
    print("z좌표 무시하고 계산")
    k = np.reshape(landmark_datas, (-1,3))
    k = k * (1,1,0)

landmark_datas = np.reshape(landmark_datas, (-1,21,3))

for landmark_data in landmark_datas:
    idx = finger_idx["Thumb"]
    v1 = landmark_data[idx[0]] - landmark_data[idx[1]]
    v2 = landmark_data[idx[2]] - landmark_data[idx[1]]
    v3 = landmark_data[idx[1]] - landmark_data[idx[2]]
    v4 = landmark_data[idx[3]] - landmark_data[idx[2]]

    theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )
    theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v3*v4) / ( math.dist(v3,(0,0,0)) *
math.dist(v4,(0,0,0)) + 1e-10 ) ) ) )

    idx = finger_idx["Index_Finger"]
    v1 = landmark_data[idx[0]] - landmark_data[idx[1]]
    v2 = landmark_data[idx[2]] - landmark_data[idx[1]]
    v3 = landmark_data[idx[1]] - landmark_data[idx[2]]
    v4 = landmark_data[idx[3]] - landmark_data[idx[2]]

    theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )
    theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v3*v4) / ( math.dist(v3,(0,0,0)) *
math.dist(v4,(0,0,0)) + 1e-10 ) ) ) )

    idx = finger_idx["Middle_Finger"]
    v1 = landmark_data[idx[0]] - landmark_data[idx[1]]
    v2 = landmark_data[idx[2]] - landmark_data[idx[1]]
    v3 = landmark_data[idx[1]] - landmark_data[idx[2]]
    v4 = landmark_data[idx[3]] - landmark_data[idx[2]]

#    print(np.rad2deg(np.arccos(np.sum(v2 * v1)/math.dist(v1, (0,0,0))*math.dist(v2, (0,0,0)))))
    theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )
    theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v3*v4) / ( math.dist(v3,(0,0,0)) *
math.dist(v4,(0,0,0)) + 1e-10 ) ) ) )

```



```

idx = finger_idx["Ring_Finger"]
v1 = landmark_data[idx[0]] - landmark_data[idx[1]]
v2 = landmark_data[idx[2]] - landmark_data[idx[1]]
v3 = landmark_data[idx[1]] - landmark_data[idx[2]]
v4 = landmark_data[idx[3]] - landmark_data[idx[2]]

theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )
theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v3*v4) / ( math.dist(v3,(0,0,0)) *
math.dist(v4,(0,0,0)) + 1e-10 ) ) ) )

idx = finger_idx["Pinky"]
v1 = landmark_data[idx[0]] - landmark_data[idx[1]]
v2 = landmark_data[idx[2]] - landmark_data[idx[1]]
v3 = landmark_data[idx[1]] - landmark_data[idx[2]]
v4 = landmark_data[idx[3]] - landmark_data[idx[2]]

theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )
theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v3*v4) / ( math.dist(v3,(0,0,0)) *
math.dist(v4,(0,0,0)) + 1e-10 ) ) ) )

#개별 손가락 사이의 각도 추가!
v1 = landmark_data[2] - landmark_data[1]
v2 = landmark_data[6] - landmark_data[5]
theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )

v1 = landmark_data[6] - landmark_data[5]
v2 = landmark_data[10] - landmark_data[9]
theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )

v1 = landmark_data[10] - landmark_data[9]
v2 = landmark_data[14] - landmark_data[13]
theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )

v1 = landmark_data[14] - landmark_data[13]
v2 = landmark_data[18] - landmark_data[7]
theta = np.append(theta, np.rad2deg(np.arccos(np.sum(v1*v2) / ( math.dist(v1,(0,0,0)) *
math.dist(v2,(0,0,0)) + 1e-10 ) ) ) )

return theta.reshape(len(landmark_datas), -1)

```

```

cap = cv2.VideoCapture(0)
deg = []

idx = 0

with mp_hands.Hands(
    model_complexity=0,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5) as hands:

    while cap.isOpened():
        success, image = cap.read()
        if not success:
            print("Ignoring empty camera frame.")
            continue

        # To improve performance, optionally mark the image as not writeable to
        # pass by reference.
        image.flags.writeable = False
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        results = hands.process(image)

        # Draw the hand annotations on the image.
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        if results.multi_hand_landmarks:

            for hand_landmarks in results.multi_hand_landmarks:
                mp_drawing.draw_landmarks(
                    image,
                    hand_landmarks,
                    mp_hands.HAND_CONNECTIONS,
                    mp_drawing_styles.get_default_hand_landmarks_style(),
                    mp_drawing_styles.get_default_hand_connections_style())
                # print(hand_landmarks)

            print(results.multi_hand_landmarks[0].landmark[0].x)
            handPosition=int(results.multi_hand_landmarks[0].landmark[0].x*100)
            print(handPosition)
            arr = np.array([])

            for point in results.multi_hand_landmarks[0].landmark:
                arr = np.append(arr, [round(point.x, 2), round(point.y, 2), round(point.z, 2)])
            arr = np.reshape(arr, (-1,3))
            deg = deg_cal(arr)

```

```
y_prob = model.predict(deg)
predicted = y_prob.argmax(axis=-1)
print(predicted[0])
msg=str(int(predicted[0]) + 1)+str(handPosition)
# msg = str(int(predicted[0]) + 1)+' '+str(results.multi_hand_landmarks[0].landmark[0].x)
print(msg)
client_socket.sendall(msg.encode())
```

```
# Flip the image horizontally for a selfie-view display.
img = cv2.flip(image, 1)
```

```
cv2.imshow('MediaPipe Hands', img)
```

```
if cv2.waitKey(1) & 0xFF == 27:
    print("close")
    cv2.destroyAllWindows()
    break
```

```
cap.release()
```