

3D 회전

종류

회전 행렬(Rotation Matrix)

행렬의 크기: 3 by 3, 3자유도 회전 = 3개의 축에 대한 회전 자유도를 가지는 것
구하는 방법: 삼각형의 합동, 벡터의 내적, 선형 변환을 이용하는 방법 등

오일러각 (Euler angle) 하나의 축을 기준으로 정점이 회전

행렬의 크기: 3 by 1

roll(x축), pitch(y축), yaw(z축) 총 3개의 축 회전에 표현, x,y,z 각 축을 독립적으로 계산, 연산량 증가 (3*3*3번)

짐벌락 문제 발생 (예) 2개의 축이 겹침 => 2 자유도 회전까지만 가능)

$$R=R_z(\theta_3)R_y(\theta_2)R_x(\theta_1)$$

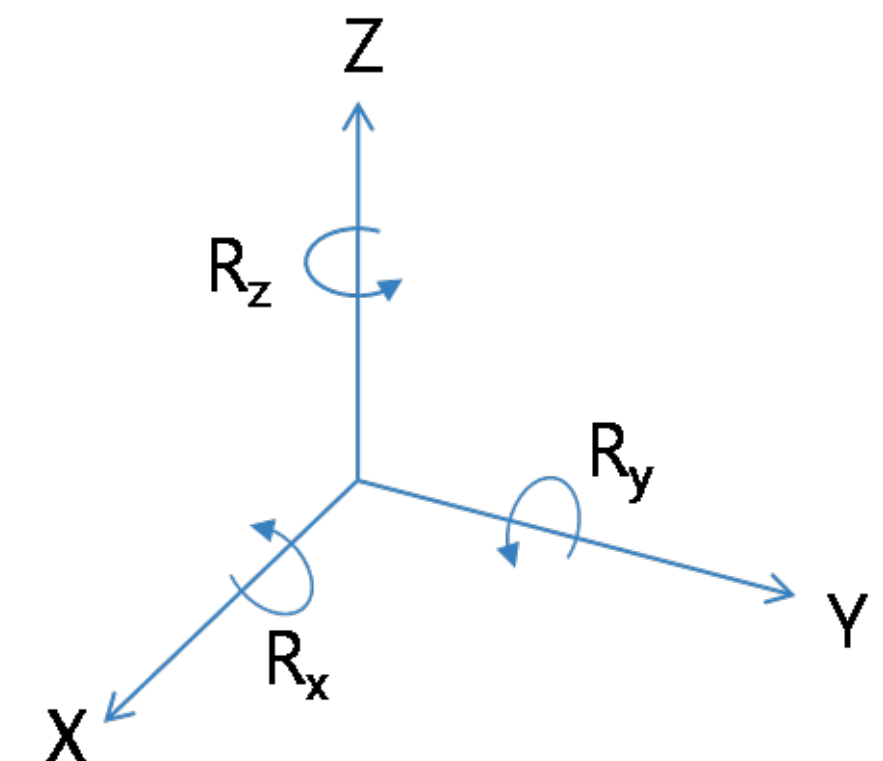
$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

쿼터니언 (Quaternion)

행렬의 크기: 4 by 1

한번에 계산 가능(종속적), x,y,z축 뿐만 아니라 임의의 축에 대한 회전을 표현 => 짐벌락 문제 해결

4차원으로 직관적x, 180도 이상인 회전 표현 불가



오른손 좌표계 (반시계)

동차좌표계

투영변환

투영변환시, 동차좌표계 사용

투영변환이란: 3차원 공간(실세계)에서의 한 점이 2차원 공간(이미지)으로 변환되는 관계
시점으로부터 방향이 중요

3차원을 동차좌표계로 표현

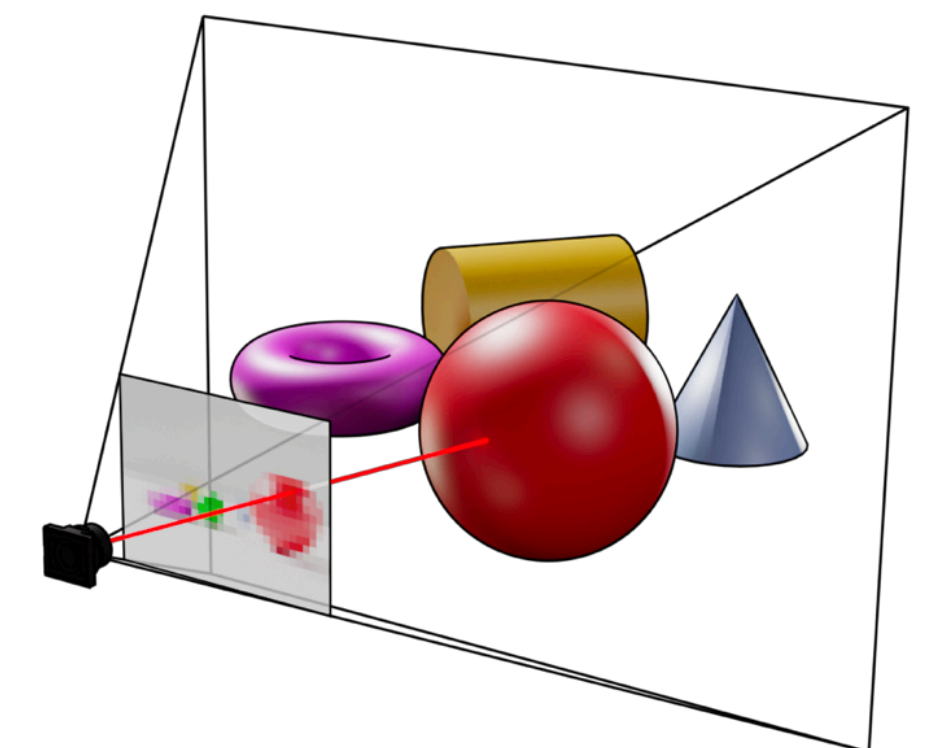
(x, y, z, w)로 표현 (w = Scale Factor)

좌표공간에서의 3d좌표를 얻어내려면 실제 3차원 좌표의 값은 (x/w, y/w, z/w) 표현할 수 있음

투영공간에서의 w: 카메라 공간에서의 카메라의 위치에서 부터 정점 사이의 거리 (월드 좌표계의 w 값은 1)

예시) 카메라 캘리브레이션 - 카메라 내부파라미터

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$



w=0일 때, 빨간 선을 가리킴

동차좌표계

벡터, 점 구분 필요없음

4주차 때 다룬 카메라 외부 파라미터로 부터 조정

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \mathbf{R} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \mathbf{t}$$

```
/* Transform 3D coordinates relative to one sensor to 3D coordinates relative to another viewpoint */
static void rs2_transform_point_to_point(float to_point[3], const struct rs2_extrinsics * extrin, const float from_point[3])
{
    to_point[0] = extrin->rotation[0] * from_point[0] + extrin->rotation[3] * from_point[1] + extrin->rotation[6] * from_point[2] + extrin->translation[0];
    to_point[1] = extrin->rotation[1] * from_point[0] + extrin->rotation[4] * from_point[1] + extrin->rotation[7] * from_point[2] + extrin->translation[1];
    to_point[2] = extrin->rotation[2] * from_point[0] + extrin->rotation[5] * from_point[1] + extrin->rotation[8] * from_point[2] + extrin->translation[2];
}
```

변환행렬: (translation vector \mathbf{t} and rotation matrix \mathbf{R})

점과 벡터에서의 변환방법이 다름. 벡터와 달리 점은 이동 변환 시, 덧셈으로 표현할 수 밖에 없음. (나머지 변환은 행렬곱)

$$\mathbf{p}' = \mathbf{M} \mathbf{p} + \vec{\mathbf{d}}$$

$$\vec{\mathbf{v}}' = \mathbf{M} \vec{\mathbf{v}}$$

동차좌표계를 통해,

점, 벡터 상관없이 동일한 연산이 가능. 즉 행렬곱만으로 모든 변환을 표현할 수 있음.

동차좌표계

n차원 사영 공간을 n+1개의 좌표로 나타내는 좌표계

2d - (x, y)를 (wx, wy, w)로, 3d - (x, y, z)를 (wx, wy, wz, w)로 표현

장점

homogeneous 좌표계를 사용하면 벡터와 점 상관없이,

(이동, 회전, 크기) 변환을 단일 행렬로 나타낼 수 있음.

행렬곱 연산만으로 표현 가능 (예시 - 뒷장)

변환식

동차좌표계 예시

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \quad \vec{\mathbf{v}} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ 0 \end{bmatrix}$$

w=1: point, w=0: vector

$$\begin{bmatrix} m_{xx} & m_{xy} & m_{xz} & d_x \\ m_{yx} & m_{yy} & m_{yz} & d_y \\ m_{zx} & m_{zy} & m_{zz} & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

행렬식: 3*3 변환행렬, 이동벡터d

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ dx & dy & dz & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

이동변환

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

크기변환

$$Rx = Rx(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

x축 회전

$$Ry = Ry(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

y축 회전

$$Rz = Rz(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

z축 회전

두 직선 사이(세 점)의 각도 구하기

3차원일 때, v1, v2 사이의 각도 구하기

$$\angle = \cos^{-1}\left(\frac{v1 \cdot v2}{\|v1\| \|v2\|}\right)$$

절차

1. 세 개의 점을 두 개의 벡터로 변환 (관심있는 점: c)

```
v1 = XYZ[a1] - XYZ[c], v2 = XYZ[a2] - XYZ[c]
```

2. v1, v2의 l2-norm 정규화 후, v1, v2를 각각 나눔

```
np.linalg.norm(값) # default: l2-norm  
v1/l2norm(v1), v2/l2norm(v2)
```

4. 2번에서 계산한 두 값을 내적

```
np.dot(값1, 값2)
```

5. arccos 값 구하기: 역삼각함수 중 하나 (x=cos y)

```
np.arccos(값)
```

6. (중요) arccos의 결과, theta 값은 radian으로 출력됨. 각도로 환산해야함. (180/PI 곱하면 됨)

```
np.rad2deg(값)
```


스켈레톤 각도 측정하기

코드 수정/추가

main.py

1. 관심있는 관절에 대한 각도 측정

‘tool/Angle’에서 get_angles함수를 통해 값을 얻음
Angle.py 수정 필요

2. 관심있는 관절에 측정한 각도를 시각화

‘tool/Visualizer3D.py’ 수정 필요

Angle.py

앞장에서 설명

데이터 타입을 dict > ndarray로 바꿈 (편의상)

main.py

```
from tool import Visualizer3D, Reader, Angle
import numpy as np

file_path = 'sample_data/sit_down_stand_up/body_joints_camXYZ.csv'

reader=Reader(file_path)
xyz_data = reader.getXYZ()
xyz_scaled_data, _ = reader.getScaledXYZ() # xyz_data[0][0]* factor

print(xyz_data.shape) # (2, 1, 5), (9, 8, 12) 확인

angle_arr = []
AnglesOfInterest = {target: Angle.BONE_RELATION[target] for target in ['ElbowLeft', 'ElbowRight', 'KneeLeft', 'KneeRight']}
for id in range(xyz_data.shape[0]):
    theta=Angle.get_angles(xyz_data[id], AnglesOfInterest)
    angle_arr.append(theta)
angle_arr = np.asarray(angle_arr)
print('angles : ', angle_arr[0][4])

visual = Visualizer3D(xyz_data=xyz_data, angle=theta, init_horizon=-65, pause_step=0.1, motion_step=1)
visual.visual_skeleton()
```

Angle.py

```
def cal_angle(line1, line2):
    line1 = line1 / np.linalg.norm(line1)
    line2 = line2 / np.linalg.norm(line2)
    angle = np.rad2deg(np.arccos(np.dot(line1, line2)))
    return angle

def get_angles(joints_camXYZ, target: dict):
    angles = []
    for a_joint, line_info in target.items():
        l1 = joints_camXYZ[line_info[0][1]] - joints_camXYZ[line_info[0][0]]
        l2 = joints_camXYZ[line_info[1][1]] - joints_camXYZ[line_info[1][0]]

        angle = cal_angle(l1, l2)
        angles.append(angle)
    return angles
```

Visualizer3D.py

코드 수정/추가

관심있는 관절

['ElbowLeft', 'ElbowRight', 'KneeLeft', 'KneeRight']

초기화 함수

파라미터 angle 추가

shape = (165, 4) # frame 번호 * 관심있는 관절 인덱스

클래스 내 visual_skeleton()

변수 theta 추가

shape = (4,) # 한 프레임의 관절 각도

3D 좌표에 text 추가

두 직선(body relation)의 교차점 위치에
theta 값을 소수 첫째자리까지만 출력

Visualizer3D.py

```
class Visualizer3D(object):
    def __init__(self, xyz_data, angle, save_path: str or None = None,
                 init_horizon=-45, init_vertical=20, x_rotation: float or int or None = None,
                 y_rotation=None, pause_step=0.2, motion_step=1):
        self.angle = angle
```

```
# 한 프레임에서의 모든 joint 좌표
x = data[frame_idx, :, 0]
y = data[frame_idx, :, 1]
z = data[frame_idx, :, 2]

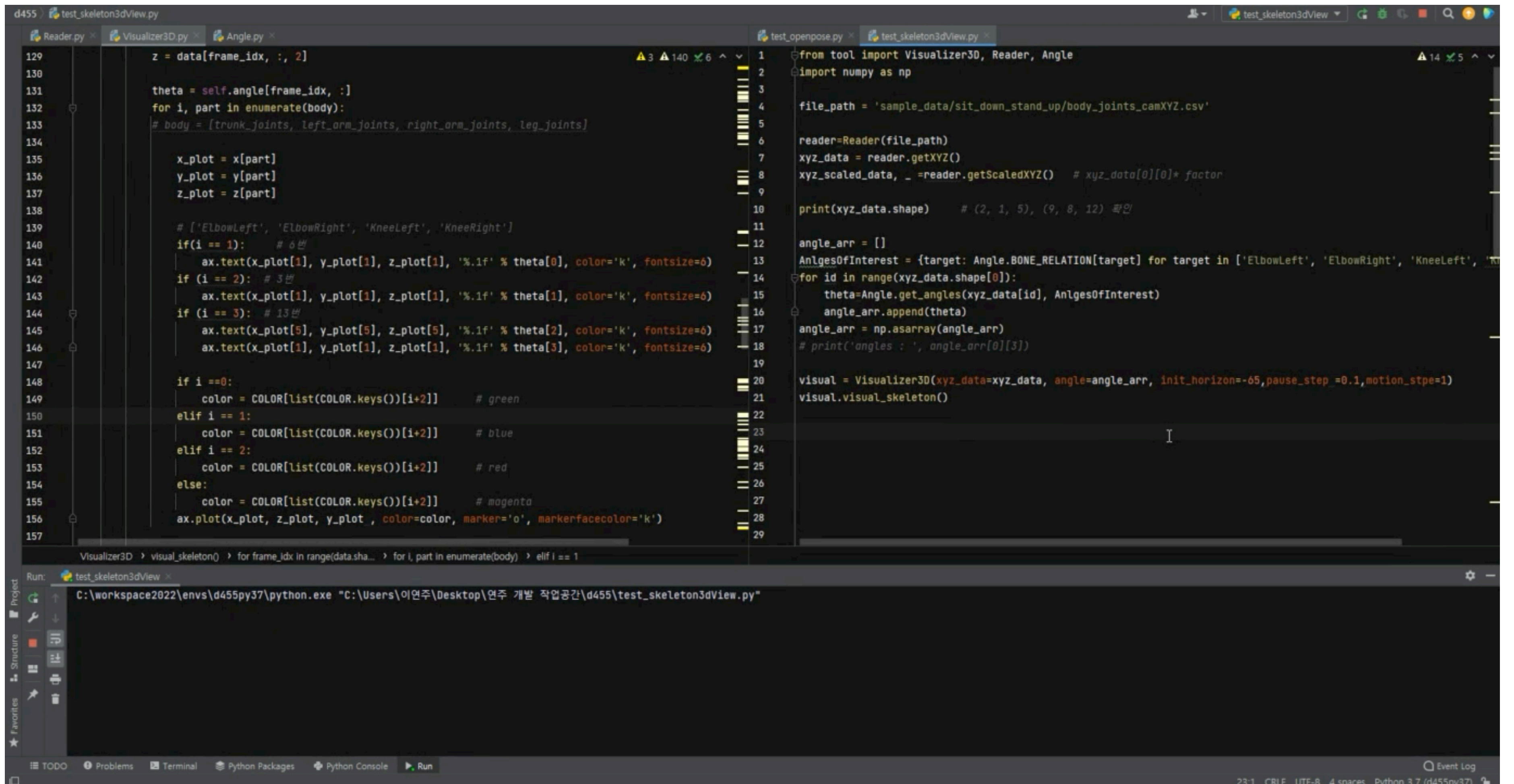
theta = self.angle[frame_idx, :]
for i, part in enumerate(body):
    # body = [trunk_joints, left_arm_joints, right_arm_joints, leg_joints]

    x_plot = x[part]
    y_plot = y[part]
    z_plot = z[part]

    # ['ElbowLeft', 'ElbowRight', 'KneeLeft', 'KneeRight']
    if(i == 1): # 6번
        ax.text(x_plot[1], y_plot[1], z_plot[1], '%.1f' % theta[0], color='red', fontsize=6)
    if (i == 2): # 3번
        ax.text(x_plot[1], y_plot[1], z_plot[1], '%.1f' % theta[1], color='red', fontsize=6)
    if (i == 3): # 13번
        ax.text(x_plot[5], y_plot[5], z_plot[5], '%.1f' % theta[2], color='red', fontsize=6)
        ax.text(x_plot[1], y_plot[1], z_plot[1], '%.1f' % theta[3], color='red', fontsize=6)

    if i == 0:
        color = COLOR[list(COLOR.keys())[i+2]] # green
    elif i == 1:
        color = COLOR[list(COLOR.keys())[i+2]] # blue
    elif i == 2:
        color = COLOR[list(COLOR.keys())[i+2]] # red
    else:
        color = COLOR[list(COLOR.keys())[i+2]] # magenta
    ax.plot(x_plot, z_plot, y_plot, color=color, marker='o', markerfacecolor='k')
```

실행 예시



```
129 z = data[frame_idx, :, 2]
130
131 theta = self.angle[frame_idx, :]
132 for i, part in enumerate(body):
133     # body = [trunk_joints, left_arm_joints, right_arm_joints, leg_joints]
134
135     x_plot = x[part]
136     y_plot = y[part]
137     z_plot = z[part]
138
139     # ['ElbowLeft', 'ElbowRight', 'KneeLeft', 'KneeRight']
140     if(i == 1): # 6번
141         ax.text(x_plot[1], y_plot[1], z_plot[1], '%.1f' % theta[0], color='k', fontsize=6)
142     if (i == 2): # 3번
143         ax.text(x_plot[1], y_plot[1], z_plot[1], '%.1f' % theta[1], color='k', fontsize=6)
144     if (i == 3): # 13번
145         ax.text(x_plot[5], y_plot[5], z_plot[5], '%.1f' % theta[2], color='k', fontsize=6)
146         ax.text(x_plot[1], y_plot[1], z_plot[1], '%.1f' % theta[3], color='k', fontsize=6)
147
148     if i == 0:
149         color = COLOR[list(COLOR.keys())[i+2]] # green
150     elif i == 1:
151         color = COLOR[list(COLOR.keys())[i+2]] # blue
152     elif i == 2:
153         color = COLOR[list(COLOR.keys())[i+2]] # red
154     else:
155         color = COLOR[list(COLOR.keys())[i+2]] # magenta
156     ax.plot(x_plot, z_plot, y_plot, color=color, marker='o', markerfacecolor='k')
157
```

```
1 from tool import Visualizer3D, Reader, Angle
2 import numpy as np
3
4 file_path = 'sample_data/sit_down_stand_up/body_joints_camXYZ.csv'
5
6 reader=Reader(file_path)
7 xyz_data = reader.getXYZ()
8 xyz_scaled_data, _ =reader.getScaledXYZ() # xyz_data[0][0]* factor
9
10 print(xyz_data.shape) # (2, 1, 5), (9, 8, 12) 확인
11
12 angle_arr = []
13 AnlgesOfInterest = {target: Angle.BONE_RELATION[target] for target in ['ElbowLeft', 'ElbowRight', 'KneeLeft', 'KneeRight']}
14 for id in range(xyz_data.shape[0]):
15     theta=Angle.get_angles(xyz_data[id], AnlgesOfInterest)
16     angle_arr.append(theta)
17 angle_arr = np.asarray(angle_arr)
18 # print('angles : ', angle_arr[0][3])
19
20 visual = Visualizer3D(xyz_data=xyz_data, angle=angle_arr, init_horizon=-65,pause_step=0.1,motion_stpe=1)
21 visual.visual_skeleton()
22
23
24
25
26
27
28
29
```

Run: test_skeleton3dView x

C:\workspace2022\envs\d455py37\python.exe "C:\Users\이연주\Desktop\연주 개발 작업공간\d455\test_skeleton3dView.py"

23:1 CRLF UTF-8 4 spaces Python 3.7 (d455py37)