HW #5 - Rigid 변환 구하기

GitHub Code link

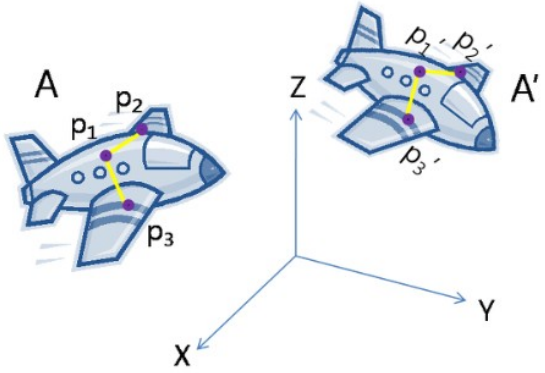
Mathvision22
02221081 황지현

HW5. Rigid 변환 구하기

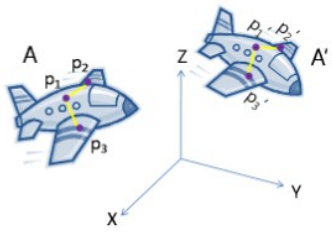
- 3D 공간에서 rigid 변환에 의해 물체의 위치가 A에서 A'으로 변하였다
  - 이 때, A의 세 점 p1, p2, p3가 A'에서 p1', p2', p3'으로 이동하였을 때, A에서의 물체 위의 임의의 점 p=(x,y,z)를 A'에서의 p'=(x',y',z')로 이동시키는 변환을 구하시오
  - 이 변환을 이용하여 p4가 p4'로 이동하는지 확인하시오. 그리고, 이를 이용하여 p5가 이동한 지점을 구하시오. 단, p1, p2, p3, p4, p5, p1', p2', p3', p4'은 다음과 같음
  - 리포트 + 검증코드(GitHub) 제출

p1 = (-0.500000, 0.000000, 2.121320)
p2 = (0.500000, 0.000000, 2.121320)
p3 = (0.500000, -0.707107, 2.828427)
p4 = (0.500000, 0.707107, 2.828427)
p5 = (1, 1, 1)

p1' = (1.363005, -0.427130, 2.339082)
p2' = (1.748084, 0.437983, 2.017688)
p3' = (2.636461, 0.184843, 2.400710)
p4' = (1.4981, 0.8710, 2.8837)
p5' = ?



- Matlab이나 파이썬, C++ 등 프로그램 툴을 이용하여 수치 계산
- 결과화면 캡쳐 (리포트에 포함)



$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R_2 R_1 \left(\begin{bmatrix} x \\ y \\ z \end{bmatrix} - \mathbf{p}_1 \right) + \mathbf{p}_1'$$

OVERALL
p_1이 원점이 되도록 A를 평행이동 후 A'과 동일하게 방향을 맞추어 p_1'으로 평행이동. 방향을 맞출 때는 먼저 평면$p_1p_2p_3$의 법선벡터(h)와 평면$p_1'p_2'p_3'$의 법선벡터(h')가 일치되도록 회전(R_1)시킨 후, $R_1(p_1p_3)$과 $p_1'p_3'$이 일치되도록 회전(R_2)

회전변환 R_1
h를 h'으로 보내기 위한 회전변환

회전변환 R_2
h'을 회전축으로 하고 벡터 $R_1(p_3-p_1)$를 $p_3'-p_1'$로 보내는 회전변환

참고) 벡터를 회전축으로 하는 회전변환
단위벡터 $u=(u_x, u_y, u_z)$를 회전축으로 하는 회전변환

$$R_u(\theta) = \begin{bmatrix} \cos\theta + u_x^2(1-\cos\theta) & u_x u_y(1-\cos\theta) - u_z \sin\theta & u_x u_z(1-\cos\theta) + u_y \sin\theta \\ u_y u_x(1-\cos\theta) + u_z \sin\theta & \cos\theta + u_y^2(1-\cos\theta) & u_y u_z(1-\cos\theta) - u_x \sin\theta \\ u_z u_x(1-\cos\theta) - u_y \sin\theta & u_z u_y(1-\cos\theta) + u_x \sin\theta & \cos\theta + u_z^2(1-\cos\theta) \end{bmatrix}$$

How to solve this problem

- Rotation angle between 2 Vecs

```
def roatationTheta(h1, h2):
    return np.arccos(np.dot(h1,h2)/(np.linalg.norm(h1)*np.linalg.norm(h2)))
```

- Rotation matrix

```
def rotationMatrix(vec1, vec2):

    H = norm_V(vec1, vec2)

    theta = roatationTheta(vec1, vec2)

    cos = np.cos(theta)
    sin = np.sin(theta)
    ux, uy, uz = H

    R = np.array([(cos + (ux ** 2) * (1 - cos), ux * uy * (1 - cos) - uz * sin, ux * uz * (1 - cos) + uy * sin),
                  (uy * ux * (1 - cos) + uz * sin, cos + (uy ** 2) * (1 - cos), uy * uz * (1 - cos) - ux * sin),
                  (uz * ux * (1 - cos) - uy * sin, uz * uy * (1 - cos) + ux * sin, cos + (uz ** 2) * (1 - cos))])
    return R
```

- Rotation

```
def rotationPos(rotatePos, originPos, comparePos, r1, r2=None):
    originPos = rotatePos -originPos
    rotationVec = r2 @ r1 @ originPos if r2 is not None else r1 @ originPos

    return rotationVec + comparePos
```

- 주어진 좌표값을 입력한 후, AA'의 평면 법선 벡터를 구한다

```
originH = np.cross((originPos[1]-originPos[0]),
                   (originPos[2]-originPos[0]))
compareH = np.cross((comparePos[1]-comparePos[0]),
                    (comparePos[2]-comparePos[0]))
```

- Find the angle & Matrix

```
theta = roatationTheta(originH, compareH)
r1 = rotationMatrix(originH, compareH)
```

- Angle A (P1, P3) -> A' (P1', P3')

```
r1_p1p3 = r1@(originPos[2]-originPos[0])
p1p3 = comparePos[2]-comparePos[0]
r2 = rotationMatrix(r1_p1p3, p1p3)


originP4 = np.array((0.500000, 0.707107, 2.828427))
compareP4 = np.array((1.498100, 0.871000, 2.883700))
```

Answer

- P4 가 제대로 P4'로 변환되는지 확인하기

```
rotationP4 = rotationPos(originP4, originPos[0], comparePos[0], r1, r2)
print(f"P4 : {rotationP4}")
print(f"P4' : {compareP4}")
```

```
P4 : [1.49808397 0.8709958  2.8837137 ]
P4' : [1.4981 0.871  2.8837]
```

- Find P5'

```
originP5 = np.array((1.000000, 1.000000, 1.000000))
rotationP5 = rotationPos(originP5, originPos[0], comparePos[0], r1, r2)
print(f"P5'의 좌표 : {rotationP5}")
```

```
P5'의 좌표 : [0.62950355 1.21310207 1.20816515]
```

<참조 >

https://www.geeksforgeeks.org/rotate-matrix-elements/
https://en.wikipedia.org/wiki/Rotation_matrix
https://darkpgmr.tistory.com/122