보행_1 일차_과제보고서

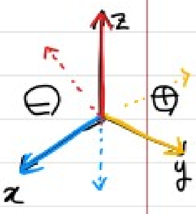
휴머노이팀 19 기 예비단원 구도연

HW_001

1. X, Y, Z 각각의 3 차원 상의 회전행렬 유도 과정 서술

보행 1일차

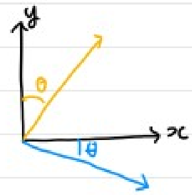
HW_001



※ 반시계 방향 회전시, 원래 축의 반대 방향에 음수붙.

i) Z축 기준으로 회전

< 2차원 >



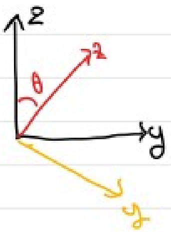
$x' = x\cos\theta - y\sin\theta$

$y' = x\sin\theta + y\cos\theta$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{aligned} R_z = \begin{bmatrix} x \\ y \\ z \end{bmatrix} &= R_z\begin{bmatrix} x \\ 0 \\ 0 \end{bmatrix} + R_z\begin{bmatrix} 0 \\ y \\ 0 \end{bmatrix} + R_z\begin{bmatrix} 0 \\ 0 \\ z \end{bmatrix} \\ &= R_z\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}x + R_z\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}y + R_z\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}z \\ &= \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix}x + \begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix}y + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}z \\ &= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} \end{aligned}$$

ii) X축 기준 회전



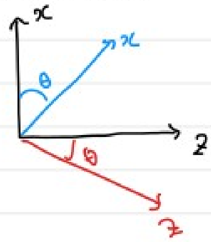
$z' = y\sin\theta + z\cos\theta$

$y' = y\cos\theta - z\sin\theta$

$$\begin{bmatrix} y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix}$$

$$\begin{aligned} R_x &= R_x\begin{bmatrix} x \\ 0 \\ 0 \end{bmatrix} + R_x\begin{bmatrix} 0 \\ y \\ 0 \end{bmatrix} + R_x\begin{bmatrix} 0 \\ 0 \\ z \end{bmatrix} \\ &= R_x\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}x + R_x\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}y + R_x\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}z \\ &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}x + \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix}y + \begin{bmatrix} -\sin\theta \\ \cos\theta \\ 1 \end{bmatrix}z \\ &= \begin{bmatrix} 1 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} \end{aligned}$$

iii) Y축 기준 회전



$x' = x\cos\theta + z\sin\theta$

$z' = -x\sin\theta + z\cos\theta$

$$\begin{bmatrix} x' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix}$$

$$\begin{aligned} R_y &= R_y\begin{bmatrix} x \\ 0 \\ 0 \end{bmatrix} + R_y\begin{bmatrix} 0 \\ y \\ 0 \end{bmatrix} + R_y\begin{bmatrix} 0 \\ 0 \\ z \end{bmatrix} \\ &= R_y\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}x + R_y\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}y + R_y\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}z \\ &= \begin{bmatrix} \cos\theta \\ 0 \\ -\sin\theta \end{bmatrix}x + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}y + \begin{bmatrix} \sin\theta \\ 0 \\ \cos\theta \end{bmatrix}z \\ &= \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} \end{aligned}$$

$\therefore$

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

$$R_y(\phi) = \begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix}$$

$$R_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2. 짐벌락 현상

1) 짐벌락 현상{key word: 행렬식, 역행렬, rank)

짐벌락이란 오일러 각도를 사용하여 같은 방향으로 오브젝트의 두 회전축이 겹치는 현상으로, 짐벌은 단일 축으로 물체가 회전하도록 중심축을 가진 구조물이 단일축으로 roll, pitch, yaw 세 개를 가진다. (이때, 각가의 고리는 본인이 가진 축을 기주으로만 회전함.)

두 번째 회전축이 돔에 따라서 첫 번째 회전축은 사만히 있는데 세 번째 회전축이 따라 돌아 첫 번째 회전축과 세 번째 회전축이 겹쳐버리지 않게 하기 위해서 생긴다.

1-1) 세 축이 종속적인 이유.

오일러 각에서 회전 자체를 이 세 축으로 나누어서 계산하기 때문.

예를들어 수학적으로 표현된 강체를 돌리기 위해서 세 축으로 회전 방향을 디지털 화 시겼고, x 에 대해서 회전하고 y 에 대해서 회전하고 z 에 대해서 회전시켰을때, a 는 x 에 대해서 회전시키면 이미 x 로 회전된 a' 의 상태이고 우리는 a 에 대해서 y 축으로 회전하는 것이 아니라 a' 에 대해서 y 축으로 회전하는 것이기 때문에 a"가 된다. 그리고 마지막으로 z 에 대해서 회전시킬 때에는 이미 우리가 알던 a 가 아니라 a"에 대해서 회전시키는 것이다. 또한 y 축으로 돌리 차례가 되엇을때에는 이미 x 로 돌아간 상태로 두 축에 대한 계산이 독립적일 수가 없다.

회전 변환 행렬을 곱할 때에 x 에 대한 회전 행렬 Rx 가 있고 y 에 대한 회전행렬 Ry 가 있고 z 에 대한 회전핼렬 Rz 가 있을때에 x 라는 물체를 회전시킨다면 RzRxRy 와 같이 계산을 할 수있다.

(계산 과정은 아래의 변환 행렬과, 행렬의 곱셈을 참고한다.)

이때, 행렬이 곱해지면 의존적으로 변하기 때문에 세 축이 아무리 독립적으로 돌리려고 하여도 그렇게 할 수 가 없는 이유이다.

또한 세 축에 대한 회전 때문에 발생하는 짐벌락 현상을 피하기 위해서는 특정한 축에 대한 회전을 시도하거나, 쿼터니언을 통해 회전시키면 된다.

이때, 특정한 축에 대해 회전하려면 일단 본인이 원하는 특정한 축을 기저의 어떤 한 축에 맞추는 회전변환 R 을 시도하고, 맞춘 기저에 대해서 원하는 만큼 회전시키고 처음 시도하는 회전변환 R 에 역행렬을 곱해주면 된다.

+ roll 은 물체의 x 축기준으로 회전 반경
+ pitch 는 물체의 y 축 기준으로 회전 반경
+ yow 는 물체의 z 기준으로 회전 반경

+쿼터니언은 (x, y, z, w) 4 차원 벡터로 세 개의 축을 동시에 회전시켜 축에 대한 종속됸계를 없앰으로 짐벌락의 문제를 방지한다.

2) 변환행렬.

회전 변환을 나타내기 위한 행렬로 각 축 방향의 회전 각도에 대하여 표현이 가능하다.

1) x축을 회전축으로 하여 회전할 때의 변환 행렬	2) y축을 회전축으로 하여 회전할 때의 변환 행렬	3) z축을 회전축으로 하여 회전할 때의 변환 행렬
$A_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix}$	$A_y = \begin{bmatrix} \cos\theta_x & 0 & \sin\theta_x \\ 0 & 1 & 0 \\ -\sin\theta_x & 0 & \cos\theta_x \end{bmatrix}$	$A_z = \begin{bmatrix} \cos\theta_x & -\sin\theta_x & 0 \\ \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 1 \end{bmatrix}$

3) 오일러 각

3 차원 공간에서 물체가 놓인 방향을 3 개의 각을 사용해 표시하는 방법.

강체가 놓인 자세를 표현하기 위해 나타내는 3 개의 서로 수직인 x, y, z 축 각도로 표현하는 방법이다.

오일러 각의 특징은 3 차원 공간의 회전을 지정할 때 직관적인 인터페이스를 제공한다는점이다.

표준기저 벡터를 회전축으로 사용하기 때문에 설계하기 용이하다.

때문에 적은 용량으로 3 차원 공간의 회전 정보를 기록할 수 있다.

2. 동차 변환 행렬을 통한 DH-파라미터 분

1) DH 파라미터

로봇 관전과 링크 간의 상대적인 위치와 방향을 설명하기 위한 방법으로 각 관절의 회전 및 평행 이동을 4 개의 매개변수로 설명함. 4 개의 매개변수는 다음과 같다.

ai: 링크 길이 (두 축 사이의 거리)

αi: 링크의 꼬임 각도 (회전축 사이의 각도)

di: 링크 편심 거리 (회전축을 따라 이동하는 거리)

θi: 조인트 각도 (회전축을 기준으로 회전하는 각도)

쉽게 이야기하면 a,α는 x축에서의 이동, 회전, d,θ는 z축에서의 이동, 회전임.

예를 들어, 3-DOF arm 으로 생각할 때, DH-표현식은 아래와 같은 행렬식으로 나타 낼 수 있고 각 관절의 변환 행렬을 위의 식을 사용항 계산한 후, 전체 변환 행렬을 진행하면 각 관절의 변환 행렬을 연속적으로 곱해 얻을 수 있다. 이렇게 얻은 행렬은 원점에서 3-DOF 팔의 끝점까지의 변환을 나타낸다.

$$T = \begin{pmatrix} \cos(\theta) & -\sin(\theta)\cos(\alpha) & \sin(\theta)\sin(\alpha) & a\cos(\theta) \\ \sin(\theta) & \cos(\theta)\cos(\alpha) & -\cos(\theta)\sin(\alpha) & a\sin(\theta) \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

HW_002

1. [X, Y, th]로 위치 정보를 표현할 때, A, B, C 가 각각 A[3,4,45°], B[-6,7,-60°], C[10,2,135°]의 위치를 가질 때 A->B 로의 동차변환행렬과 C->B 로의 동차변환 행렬만을 가지고 A->C 로의 이동을 Eigen 으로 표현. (코드 + 결과 이미지 첨부)

<코드 첨부>

```
#include "rclcpp/rclcpp.hpp"
#include <Eigen/Dense>
#include <cmath>
#include <sstream>

// 변환행렬 생성 함수
Eigen::Matrix3d createTransformationMatrix(double x, double y, double theta) {
Eigen::Matrix3d transform = Eigen::Matrix3d::Identity();
double rad = theta * M_PI / 180.0;
transform(0, 0) = cos(rad);
transform(0, 1) = -sin(rad);
transform(1, 0) = sin(rad);
transform(1, 1) = cos(rad);
transform(0, 2) = x;
transform(1, 2) = y;
return transform;
}

int main(int argc, char * argv[]) {
rclcpp::init(argc, argv);
auto node = rclcpp::Node::make_shared("transform_node");

// A->B, C->B 변환행렬 계산
Eigen::Matrix3d T_AB = createTransformationMatrix(-6 - 3, 7 - 4, -60 - 45);
Eigen::Matrix3d T_CB = createTransformationMatrix(10 - (-6), 2 - 7, 135 + 60);

// A->C 변환행렬 계산
Eigen::Matrix3d T_AC = T_AB * T_CB.inverse();

// Eigen::IOFormat 을 사용하여 출력할 수 있는 형식으로 변환
Eigen::IOFormat fmt(Eigen::FullPrecision, Eigen::DontAlignCols, ", ", "\n", "[", "]");

// std::stringstream 을 사용하여 변환된 행렬을 문자열로 변환
std::stringstream ss;
ss << T_AC.format(fmt);
std::string result = ss.str();

// 변환된 행렬을 RCLCPP_INFO 로 출력
RCLCPP_INFO(node->get_logger(), "Transformation from A to C:\n%s", result.c_str());
```

```
rclcpp::shutdown();
return 0;
}
```

<실행 결과>

```
dy@dy-ASUS-TUF-Gaming-F17-FX707VI-FX707VI:~/colcon_ws$ ros2 run transform_exampl
e transform_node
[INFO] [1731494669.862989581] [transform_node]: Homogeneous Transformation from
A to B:
[-0.258819045102521, 0.965925826289068, -9]
[-0.965925826289068, -0.258819045102521, 3]
[0, 0, 1]
[INFO] [1731494669.863036488] [transform_node]: Homogeneous Transformation from
C to B:
[-0.965925826289068, 0.25881904510252, 16]
[-0.25881904510252, -0.965925826289068, -5]
[0, 0, 1]
[INFO] [1731494669.863043194] [transform_node]: Homogeneous Transformation from
A to C:
[0.5, -0.866025403784439, -21.3301270189222]
[0.866025403784439, 0.5, -8.35640646055102]
[0, 0, 1]
```