REPORT

3D.O.F PRR Robot Arm Calibration

학번	72210276	과목명	로보틱스
전공	기계공학과	담당교수	김태정 교수님
이름	이진우	제출일	2021.12.16

DKU 단국대학교 DANKOOK UNIVERSITY

1. 3자유도 로봇 설계

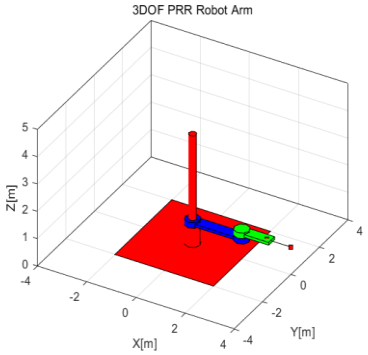

Fig. 1 3DOF PRR Robot Arm

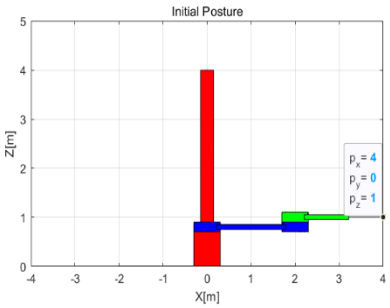

Fig. 2 로봇의 초기 종단부의 자세

그림 1 에 나타낸 로봇은 물체의 형상정보를 측정하도록 설계된 가상의 3 축 로봇이다. 이 로봇은 기지점 (0,0,0)으로부터 땅에서부터 순서대로 프리즘 관절, 회전관절, 회전관절로 연결된 PRR 직렬기구이다. 각 링크의 길이는 1 번링크부터 1m, 2m, 2m 이며, 3 번째 링크의 종단부상에 관측대상의 한점을 작업공간 내 위치로 나타낼 수 있는 측정기가 장착되어 있으며 측정기의 끝점을 붉은색 점으로 표현하였다. 각 링크들이 회전관절과 프리즘 관절로 순차적으로 연결되어 있는 이 로봇의 자유도는 3 이며, 종단부의 방위와 위치를 3 개의 관절변수 θ로 나타낼 수 있다. 주어진 관절 변수 θ로부터 표현된 종단부의 방위와 자세를 각각 $R_{st}(\theta)$ 와 $p_{st}(\theta)$ 라 하자. 각 관절 변수가 모두 0 일 때 종단부의 초기 방위와 위치는 다음 표에 작성하였다.

Table. 1 로봇의 종단부 초기 자세 및 방위정보

방위	위치
$R_{st}(0)=\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$	$p_{st}(0)=\begin{bmatrix}4&0&1\end{bmatrix}^T$

로봇팔을 구성하고 있는 관절들의 관절각이 모두 결정되었을 때, 종단부의 자세를 결정하는 식을 정기구식이라 하며 이는 4x4 동차행렬로 나타낼 수 있다. 종단부에 먼저 연결되어 있는 축부터 땅까지 순차적으로 축을 지정하여 주어진 관절각으로부터 동차행렬의 꼴로 표현한 강체변환행렬은 곱하는 것으로 정기구식을

구할 수 있다. 이때, 축의 번호는 땅에 먼저 연결된 순서를 낮은 순서로 부여하도록 약속하자. 주어진 기준자세에서 3개의 축의 강체변환행렬은 다음과 같이 구할 수 있다.

주어진 초기 종단부의 자세: $G_{\mathrm{st}}(0)=\begin{bmatrix} I & p_{\mathrm{t}} \\ 0^T & 1 \end{bmatrix}$

1 번째 축의 강체변환행렬: ${}^{a}{}_{1}G(\theta_1)=\begin{bmatrix} I & \theta_1 u \\ 0^T & 1 \end{bmatrix}$

2 번째 축의 강체변환행렬: ${}^{a}{}_{2}G(\theta_2)=\begin{bmatrix} \mathrm{R_z}(\theta_2) & (\mathrm{I}-\mathrm{R_z}(\theta_2))\,\mathrm{q}_2 \\ 0^{\mathrm{T}} & 1 \end{bmatrix}$

3 번째 축의 강체변환행렬: ${}^{a}{}_{3}G(\theta_3)=\begin{bmatrix} \mathrm{R_z}(\theta_3) & (\mathrm{I}-\mathrm{R_z}(\theta_3))\,\mathrm{q}_3 \\ 0^{\mathrm{T}} & 1 \end{bmatrix}$

주어진 관절각으로부터 구한 종단부의 자세는 다음과 같다.

$$G_{st}(\theta)={}^{a}{}_{1}G(\theta_1)\,{}^{a}{}_{2}G(\theta_2)\,{}^{a}{}_{3}G(\theta_3)\,G_{st}(0)$$

$$=\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & L_2\cos\theta_2+L_3\cos\alpha \\ \sin\alpha & \cos\alpha & 0 & L_2\sin\theta_2+L_3\cos\alpha \\ 0 & 0 & 1 & L_1+\theta_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

여기서, α는 2 번째 관절각과 3 번째 관절각의 합이다.

$$\alpha=\theta_2+\theta_3$$

종단부의 자세가 주어졌을 때, 관절변수를 결정하는 문제를 역기구학 문제이며 정기구식으로부터 종단부의 위치 $\boldsymbol{p}_{st}=\begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$와 각 링크의 길이 $\boldsymbol{L}=\begin{bmatrix} L_1 & L_2 & L_3 \end{bmatrix}^T$이 주어졌을 때 각 관절각은 다음의 식으로 구할 수 있다.

$$\theta_1 = p_z - L_1$$

$$\theta_3 = \cos^{-1}\left(\frac{p_x{}^2 + p_y{}^2 - L_2^2 - L_3^2}{2L_2L_3}\right)$$

$$\theta_2 = arg\left(\frac{1}{k_1^2 + k_2^2}\begin{bmatrix} -k_2p_x + k_1p_y \\ k_1p_x + k_2p_y \end{bmatrix}\right)$$

여기서, 각각 k_1과 k_2는

$$k_1 = L_2 + L_3\cos\theta_3$$

$$k_2 = L_3\sin\theta_3$$

이다. 붉은색 점선과 같이 X, Y, Z 좌표 값으로 주어진 경로가 입력되었을 때, 역기구식을 계산하여 각 관절각을 구하고 구해진 관절각으로부터 정기구식을 풀어 계산된 종단부의 위치가 맞는지 가시화하여 확인하였다.

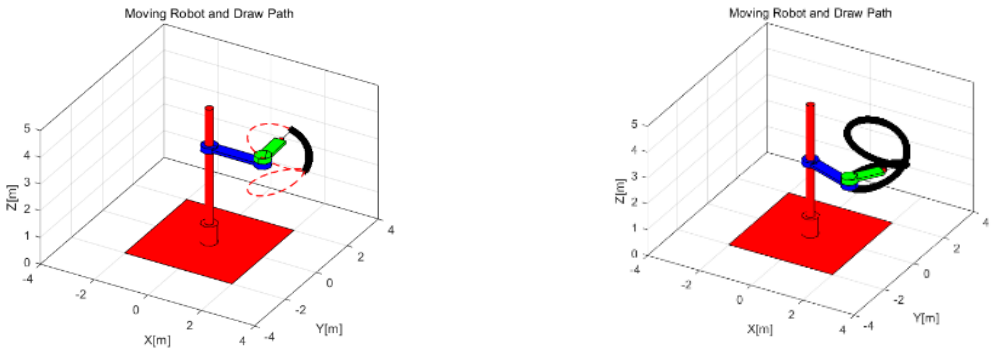

Fig. 3 정기구식을 통해 구한 종단부의 경로(3D)

2. 경로 계획

로봇의 말단장치가 관절들을 움직여 도달할 수 있는 공간을 작업공간이라 하며, 작업공간 내에서 로봇이 운동 중에 특정 자세에서 자유도를 잃는 경우가 발생하는데 이 위치를 특이점, 자세를 특이자세라 한다. 경로를 계획하기에 앞서 생성한 경로에 특이자세가 있는지 확인해야 하며 각 위치에 대한 로봇의 말단 장치의 자코비안행렬을 구하고 이를 음함수 정리(Implicit function theorem)를 이용하여 속도가 정의될 수 없는 위치인지 파악해야 한다. 종단부의 자코비안행렬은 정기구식으로부터 구할 수 있으며 설계한 로봇팔의 자코비안행렬은 다음과 같다.

$$\mathbf{w}_{st}=\begin{bmatrix}\mathbf{v}_{st}\\ \boldsymbol{\omega}_{st}\end{bmatrix}=\mathbf{J}_{st}\dot{\theta}$$

$$J_{st}=\begin{bmatrix}0 & -L_2\sin(\theta_2)-L_3\sin(\theta_2+\theta_3) & -L_3\sin(\theta_2+\theta_3)\\ 0 & L_2\cos(\theta_2)+L_3\cos(\theta_2+\theta_3) & L_3\cos(\theta_2+\theta_3)\\ 1 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 1 & 1\end{bmatrix}$$

여기서, 독립적인 1 열 및 0 벡터인 3 행과 4 행을 제외한 나머지 벡터들로 다시 자코비안행렬을 구하면

$$\begin{bmatrix}-L_2\sin(\theta_2)-L_3\sin(\theta_2+\theta_3) & -L_3\sin(\theta_2+\theta_3)\\ L_2\cos(\theta_2)+L_3\cos(\theta_2+\theta_3) & L_3\cos(\theta_2+\theta_3)\\ 1 & 1\end{bmatrix}$$

이다. 다음의 식을 만족할 때 음함수 정리(Implicit function theorem)에 따라 종단부의 속도를 정의할 수 있으며 경로를 계획하고 다음의 경로에 조건이 만족하는지 확인하였다.

$$\theta_3 \neq 0, \pi, 2\pi, \ldots, n\pi$$

설계한 PRR 로봇팔의 작업공간은 다음과 같으며,

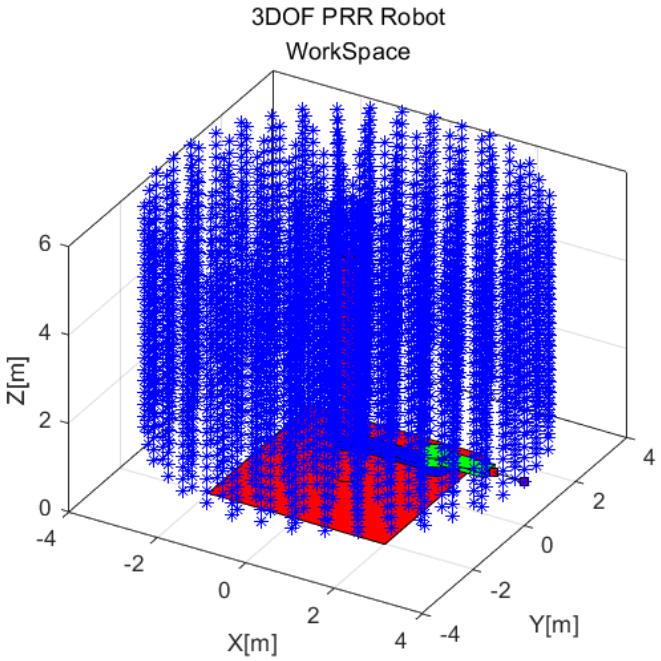

Fig. 4 PRR Robot's WorkSpace

작업공간 내 아래의 경로를 생성하고 생성된 경로에서 위에 구한 3 번째 관절각의 각도가 만족하는지 확인하였으며 최종적으로 아래와 같은 곡선을 선택하였다.

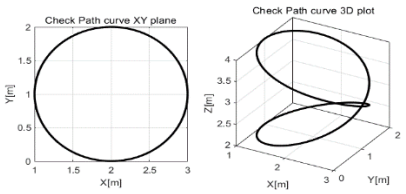

Fig. 5 종단부의 입력된 작업 경로

위 그림에 입력된 작업 경로는 다음의 t 를 매개변수로 하는 매개변수함수로부터 0 부터 $2pi$ 구간까지의 곡선으로 지정하였다.

$$x_p(t) = 2 + 2\cos 2t$$
$$y_p(t) = 1 + 2\sin 2t$$
$$z_p(t) = 3 + 2\sin t$$

3. 오차 및 보정

로봇팔의 종단부에 위치를 측정할 수 있는 센서를 부착한 뒤 종단부 한점의 위치를 측정하였다고 하자. 이때 로봇팔에 입력한 경로로부터 난수로 얻은 오차를 더하여 측정된 점을 얻고 이를 아래 그래프에 나타내었다.

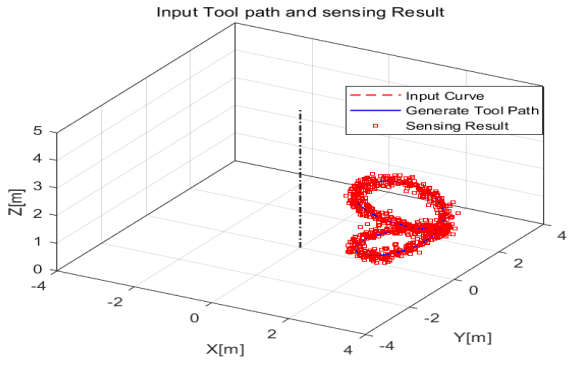

Fig. 6 입력 경로와 역기구식을 통해 얻은 생성경로 및 측정된 경로 비교

오차는 평균 0.05, 표준편차 0.05 인 정규분포의 형태를 갖도록 하였다. 정규분포를 갖도록 난수를 발생시킨 결과 오차 그래프는 아래와 같다.

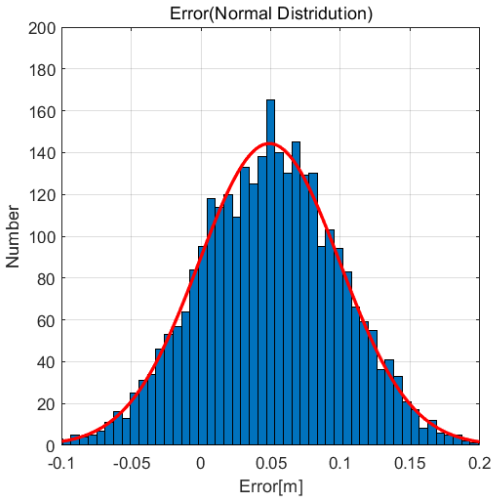

Fig. 7 오차그래프

아래와 같이 링크에 ε만큼의 오차를 가정하여 기구적으로 앞선 오차를 보정하고자 한다.

$$\left[L_1 \; L_2 \; L_3 \right]^T \rightarrow \left[L_1+\varepsilon_1 \; L_2+\varepsilon_2 \; L_3+\varepsilon_3 \right]^T$$

오차 파라미터를 적용하여 구한 종단부의 위치는 다음과 같다.

$$\begin{bmatrix} \left(L_2+\varepsilon_2\right)\cos\theta_2+\left(L_3+\varepsilon_3\right)\cos\alpha \\ \left(L_2+\varepsilon_2\right)\sin\theta_2+\left(L_3+\varepsilon_3\right)\sin\alpha \\ \left(L_1+\varepsilon_1\right)+\theta_1 \end{bmatrix}$$

이때 i 번째 위치에서 관절각 변수를

$$\theta_{\mathrm{i}}=\left[\theta_{1,\mathrm{i}} \; \theta_{2,\mathrm{i}} \; \theta_{3,\mathrm{i}} \right], \alpha_i=\theta_{2,\mathrm{i}}+\theta_{3,\mathrm{i}}$$

라 하자. i 번째 관절각에서 측정된 종단부의 위치를 $\boldsymbol{p}^*_{\mathrm{st,i}}$ 라 할 때 오차를 다음과 같이 구할 수 있다.

$$\sum_{i=1}^{N} r_i^2=(A\varepsilon-b)^T(A\varepsilon-b)$$

$$r_i^2=\left(\cos\theta_{2,i}\varepsilon_2+\cos\alpha_i\varepsilon_3+L_2\cos\theta_{2,i}+L_3\cos\alpha_i-p^*_{x,i}\right)^2+$$
$$\left(\sin\theta_{2,i}\varepsilon_2+\sin\alpha_i\varepsilon_3+L_2\sin\theta_{2,i}+L_3\sin\alpha_i-p^*_{y,i}\right)^2+$$
$$\left(\varepsilon_1+\theta_{1,i}L_1-p^*_{z,i}\right)^2$$

여기서 A, ε, b는 다음과 같다.

$$A=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \cos\theta_{3,1} \\ 0 & \cos\theta_{3,1} & 1 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \\ 0 & 1 & \cos\theta_{3,n} \\ 0 & \cos\theta_{3,n} & 1 \end{bmatrix}, \varepsilon=\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix}, b=\begin{bmatrix} p^*_{z,1}-\theta_{1,1}-L_1 \\ p^*_{x,i}\cos\theta_{2,1}+p^*_{y,i}\sin\theta_{2,1}-L_2-L_3\cos\theta_{3,1} \\ p^*_{x,i}\cos\alpha_1+p^*_{y,i}\sin\alpha_1-L_2\cos\theta_{3,1}-L_3 \\ \vdots \\ p^*_{z,n}-\theta_{1,n}-L_1 \\ p^*_{x,n}\cos\theta_{2,n}+p^*_{y,n}\sin\theta_{2,n}-L_2-L_3\cos\theta_{3,n} \\ p^*_{x,n}\cos\alpha_1+p^*_{n,i}\sin\alpha_1-L_2\cos\theta_{3,n}-L_3 \end{bmatrix}$$

위 식을 오차 파라미터에 대하여 오차를 최소화하는 최소자승법(Least Square Method)를 사용하여 오차 파라미터를 계산하였다. 오차 파라미터를 구하는 식은 다음과 같다.

$$\varepsilon=(\mathrm{A^TA})^{-1}A^Tb$$

링크의 길이를 보정한 결과는 다음과 같다.

Fig. 8 보정 전과 보정 후 링크의 길이비교

링크	보정 전 링크의 길이	보정 후 링크의 길이
L1	1 [m]	1.0493 [m]
L2	2 [m]	2.0333 [m]
L3	2 [m]	2.0566 [m]

아래 그림은 오차보정을 적용하여 같은 관절변수가 주어졌을 때 곡선의 변화를 보여주는 그림이다. 붉은색 실선은 이전에 입력한 경로이며 푸른색은 보정 후 길이로 계산한 종단부의 경로를 나타내는 곡선이다.

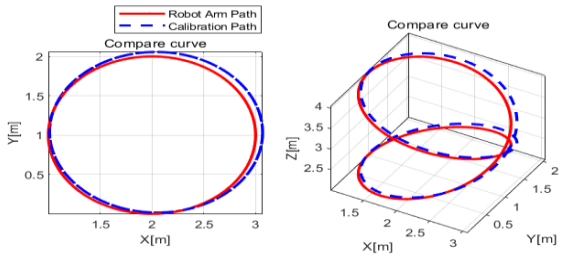

Fig. 9 보정 후 동일 관절각에 대한 곡선과 입력곡선 비교

처음 입력한 종단부의 경로를 나타내기 위해 오차보정이 적용된 로봇에 대하여 역기구식을 풀어 나타낸 관절변수의 변화를 아래 그림에 플로팅하여 비교하였다.

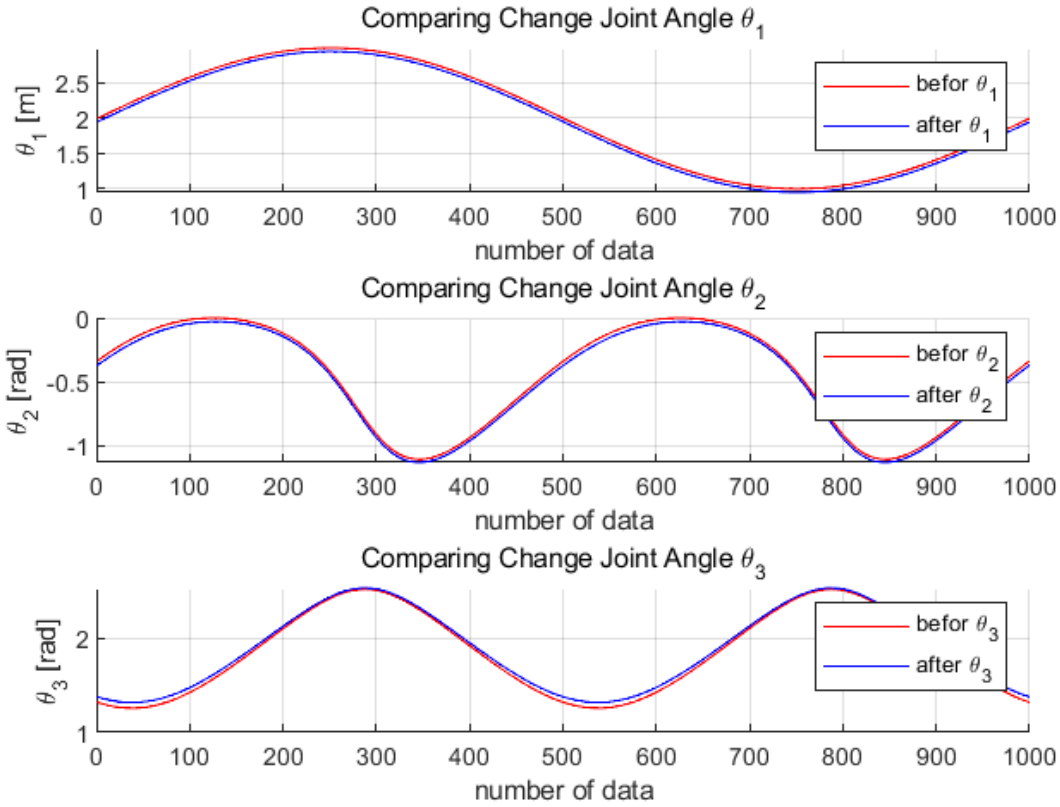

Fig. 10 동일한 입력곡선에 대한 보정 전/보정 후 관절각의 각도변화

오차가 발생한 점으로부터 최소자승법으로 보정하여 얻은 점과 보정 전의 점의 평균제곱오차(MSE)를 구하였고 결과적으로 오차가 감소한 것을 확인할 수 있다.

Table. 2 보정 전과 후 평균제곱오차 비교표

	보정 전	보정 후
MSE	0.1495	0.730

Appendix. A matlab code

[1] 실행 진입 코드 main.m

```
clc; clear all, close all
%% 2021 12 16 robotics
%  by Lee jin woo
%  CP is Center Point of Circle Center; R = Radius N is Point number
%  linkdata is [L1 L2 L3];

CP = [2,1,3]; R=1; N=1000;
linkdata = [1,2,2]; soltype = 1;
data = MyCalibration(CP,R,N,linkdata,soltype)
```

[2] MyCalibration.m (function code)

```
function data = MyCalibration(CP,R,N,linkdata,soltype)
%% 2021 12 15 CADCAM LAB Jinwoo Lee
% addpath forlder
addpath("data","body","calculate","kinematics","drawing")
% input data linkdata = [L1, L2, L3];
%           CP      = [Cx, Cy, Ho];
%           R       = Path radius;
%           N       = Number of point;
%           soltype  = 1 or 2 (solution 1 -> pi>theta3>0)
%                             (solution 2 -> 2pi>theta3>pi)
% output data Calibration Link length and Change of MSE
%% Path generate
[f0,PathData] = gen_path(CP,R,N);
%% Robot genrate
f1 = figure();
lm = [[-1,1]*sum(linkdata(2:3)),[-1,1]*sum(linkdata(2:3)),0,5*linkdata(1)];
ax1 = mysetax(f1,"3DOF PRR Robot",lm,["X[m]","Y[m]","Z[m]"],[0,0]);
PRR=Gen_PRR(ax1,linkdata,0.3,0.15,5,0.2,0.1); hold on;
%% workspace
sca=draw_workspace(ax1,PRR);
fprintf("Please Wait then clear Workspace\n"); pause(5);
delete(sca); ax1.Subtitle =[];
%%  Get Joint Angle
jointdata = get_joint(linkdata,PathData,soltype);
add_pathcurve(ax1,PathData)
moving_robot(ax1,PRR,jointdata);
f3 = figure();
ax2 = mysetax(f3,"Draw Path",lm,["X[m]","Y[m]","Z[m]"],[30,30]);
legend_list=draw_curve(ax2,linkdata,jointdata,PathData,"on");
err = get_Error(0.05,0.05,N,'on');
errcurve = err+PathData;
get_joint(linkdata,errcurve,soltype); %check Jacobian;
legend_list(end+1) = add_ecurve(ax2,errcurve);
ax2.Title=text(0.5,0.5,'Input Tool path and sensing Result');
legend(legend_list,"Input Curve","Generate Tool Path","Sensing Result");
add_ecurve(ax1,errcurve);
eps = Least_square(linkdata,jointdata,errcurve);
fprintf("Get Error Vector %f: \n",eps);
%% Calibration
f4 = figure();
calibrationlinkdata = linkdata + eps';
compare_curve(f4,jointdata,linkdata,calibrationlinkdata);
% ------------------------------plot theta---------------------------------
```

```
f5 = figure();
calibrationJointdata=get_joint(calibrationlinkdata,PathData,soltype);
compare_joint(f5,jointdata,calibrationJointdata);
[preMSE,postMSE]=compare_MSE(jointdata,linkdata,calibrationlinkdata,errcurve);
%% output data
data.calibrationlinkdata=calibrationlinkdata;
data.preMSE=preMSE;
data.postMSE=postMSE;
end
```

[3] 가상 경로 생성 코드

```
function [f,pathcurvepoint] = gen_path(CP,R,N,varargin)
% x->x(t), y->y(t) z->z(t)
% CP is Circle Center Point -> [Cx,Cy,Zo], R;
% Parameter -> t
R=abs(R);
pathcurvepoint = zeros(N,3);
t = linspace(0,4*pi,N);

for k = 1:N
    pathcurvepoint(k,:) = [CP(1)+R*cos(t(k)),CP(2)+R*sin(t(k)),...
        CP(3)+R*sin(0.5*t(k))];
end

f=figure();
lm=[-1,1,-1,1,-1,1]*R+[CP(1),CP(1),CP(2),CP(2),CP(3),CP(3)];
ax1 = mysetax(f,'Check Path curve XY plane',lm,["X[m]","Y[m]","Z[m]"],[0,90]);
ax2 = mysetax(f,'Check Path curve 3D plot',lm,["X[m]","Y[m]","Z[m]"],[30,30]);
path_line1 = line(ax1); path_line2 = line(ax2);
subplot(1,2,1,ax1); subplot(1,2,2,ax2);
set(path_line1,'xdata',pathcurvepoint(:,1),'ydata',pathcurvepoint(:,2),...
    'zdata',pathcurvepoint(:,3),'Linestyle','-','color','black',...
    'linewidth',2);
set(path_line2,'xdata',pathcurvepoint(:,1),'ydata',pathcurvepoint(:,2),...
    'zdata',pathcurvepoint(:,3),'Linestyle','-','color','black',...
    'linewidth',2);
end
```

[4] Body 관련 matlab 코드

[4.1] Gen_PRR code: Patch를 이용하여 로봇팔 생성

```
function PRR=Gen_PRR(ax,link_data,ro,ri,H,h,w)
% function information
% input ax        : draw space (vareiable)
%       link_data : [L1,L2,L3] -> pt = [L1,L2,L3];
%       ro        : out radius (bigger than ri)
%       ri        : in radius (holes or pin size)
%       H         : Base's rod Height[m]
%       h         : pin's rod Height[m]
%       w         : pin - pin connect reactanguler size [m]
% ============================setting Link=====================================
L1 = link_data(1); L2 = link_data(2); L3 = link_data(3);
%ax.View = [0,0];
%ax.Title=text(0.5,0.5,'Initial Posture');
% ============================gen Base=========================================
react = [1,1]*(L2+L3);
Base = gen_Base(ax,react,ro,ri,H,L1-1.5*h);
% ============================gen Link=========================================
p1 = [0,0,L1-h];
Link2 = gen_Link2(ax,p1,L2,ro,ri,w,h);
% ============================gen Link=========================================
p2 = [L2,0,L1];
[Link3,pt] = gen_Link3(ax,p2,L3,ro,ri,w,h);

PRR.Base = Base;
PRR.Link2 = Link2;
PRR.Link3 = Link3;

PRR.origion = [0,0,0];
PRR.p1 = p1;
PRR.p2 = p2;
PRR.pt = pt;
end
```

[4.2] gen_Base: 땅과 연결된 프리즘 링크 생성

```
function Base = gen_Base(ax,react,ro,ri,H,h)
% function information
% input
%       react     : [base,height]
%       ro        : Link1 rod radius
%       ri        : base rod radius
%       H         : base rod Height
%       h         : base rod height
% output
%       Base      : One bord(reactanguler) and One rod, one Link1(rod)
%                 : Origin is global origin (0,0,0) and bord center same
%                 : rod height is H and radius ri
%                 : Link1 height is h radius is ro
%=========================function=========================================
%% 사각형 생성
react = abs(react); react = react+0.01; %0 방지;
Box = [-0.5*react(1),-0.5*react(2);...
    0.5*react(1),-0.5*react(2);...
    0.5*react(1),0.5*react(2);...
    -0.5*react(1),0.5*react(2)];
%% 로드 원 정보
theta = 0:pi/4:2*pi-pi/4;
CO = zeros(8,2);
for k = 1:length(theta)
    CO(k,:) = [ro*cos(theta(k)),ro*sin(theta(k))];
end
Ci = zeros(8,2);
for k = 1:length(theta)
    Ci(k,:) = [ri*cos(theta(k)),ri*sin(theta(k))];
end
%% Generate Vertex
Vertices = zeros(36,3);
Vertices(1:4,:) = [Box,zeros(4,1)];
Vertices(5:20,:) = [[CO;CO],[zeros(8,1);ones(8,1)*h]];
Vertices(21:36,:) = [[Ci;Ci],[ones(8,1)*h;ones(8,1)*H]];

%% Generate Face information
Faces = zeros(7,11); Faces(:,:) = nan; %빈공간은 nan 처리.
Faces(1,1:5) = [1:4,1];
Faces(2,1:9) = [13:20,13];
Faces(3,:)   = [5:9,17:-1:13,5];
Faces(4,:)   = [9:12,5,13,20:-1:17,9];
Faces(5,1:9) = Faces(2,1:9) + 16;
Faces(6:7,:) = Faces(3:4,:) + 16;

Base.body=patch(ax,"Faces",Faces,'Vertices',Vertices,'FaceColor','red');
end
```

[4.3] Link2 생성

```
function Link2 = gen_Link2(ax,po,l,ro,ri,w,h)
% function information
% input ax        : draw space (vareiable)
%       po        : po(local origin);
%       l         : length
```

```
%       ro        : out radius (bigger than ri)
%       ri        : in radius (holes or pin size)
%       w         : pin - pin connect reactanguler size [m]
%       h         : pin - pin connect reactanguler  size [m]
%% 원 정보
theta = 0:pi/4:2*pi-pi/4;
CO = zeros(8,2);
po = reshape(po,1,3);

for k = 1:length(theta)
    CO(k,:) = [ro*cos(theta(k)),ro*sin(theta(k))]+po(1:2);
end
Ci = zeros(8,2);
for k = 1:length(theta)
    Ci(k,:) = [ri*cos(theta(k)),ri*sin(theta(k))]+po(1:2);
end
%% Generate Vertex
Vertices=zeros(92,3);
Vertices(1:16,:) = [[CO;CO],[ones(8,1)*(po(3)-0.5*h);ones(8,1)*(po(3)+0.5*h)]];
Vertices(17:32,:)= [[Ci;Ci],[ones(8,1)*(po(3)-0.5*h);ones(8,1)*(po(3)+0.5*h)]];
Vertices(33:48,:)= [Vertices(1:16,1)+l,Vertices(1:16,2:3)];
Vertices(49:64,:)= [Vertices(17:32,1)+l,Vertices(17:32,2:3)];
Vertices(65:80,:)= [Vertices(49:64,1:2),Vertices(49:64,3)+h];
b = ([-1 -1 -1 1 1 1]*(0.5*w)+po(3))';
Vertices(81:86,:)= [Vertices([8,1,2,10,9,16],1:2),b];
Vertices(87:92,:)= [Vertices([46:-1:44,36:38],1:2),flip(b)];


%% Genrate Faces
Faces = zeros(20,13); Faces(:,:) = nan;
Faces(1,1:11) = [1:5,21:-1:17,1];
Faces(2,1:11) = [5:8,1,17,24:-1:21,5];
Faces(3,1:11) = Faces(1,1:11)+8;
Faces(4,1:11) = Faces(2,1:11)+8;
Faces(5,1:11) = Faces(1,1:11)+32;
Faces(6,1:11) = Faces(2,1:11)+32;
Faces(7,1:11) = Faces(5,1:11)+8;
Faces(8,1:11) = Faces(6,1:11)+8;
Faces(9,1:11) = [1:5,13:-1:9,1];
Faces(10,1:11) = [5:8,1,9,16:-1:13,5];
Faces(11,1:11) = Faces(9,1:11)+16;
Faces(12,1:11) = Faces(10,1:11)+16;
Faces(13,1:11) = Faces(9,1:11)+32;
Faces(14,1:11) = Faces(10,1:11)+32;
Faces(15,1:11) = Faces(13,1:11)+8;
Faces(16,1:11) = Faces(14,1:11)+8;
Faces(17,1:5) = [81 86 87 92 81];
Faces(18,1:5)= [83 84 89 90 83];
Faces(19,1:7)= [84:89,84];
Faces(20,1:7)= [81 82 83 90 91 92 81];

Link2.body=patch(ax,"Faces",Faces,'Vertices',Vertices,'FaceColor','blue');
end
```

[4.4] Link2 생성(종단부)

```
function [Link3,pt] = gen_Link3(ax,po,l,ro,ri,w,h)
% function information
% input ax        : draw space (vareiable)
%       po        : po(local origin);
%       l         : length
%       ro        : out radius (bigger than ri)
%       ri        : in radius (holes or pin size)
%       w         : pin - pin connect reactanguler size [m]
%       h         : pin - pin connect reactanguler  size [m]
% output pt       : 종단부 point
%% 원 정보
theta = 0:pi/4:2*pi-pi/4;
CO = zeros(8,2);
po = reshape(po,1,3);

for k = 1:length(theta)
    CO(k,:) = [ro*cos(theta(k)),ro*sin(theta(k))]+po(1:2);
end
Ci = zeros(8,2);
for k = 1:length(theta)
    Ci(k,:) = [ri*cos(theta(k)),ri*sin(theta(k))]+po(1:2);
end
%% Generate Vertex

Vertices=zeros(44,3);
Vertices(1:16,:) = [[CO;CO],[ones(8,1)*(po(3)-0.5*h);ones(8,1)*(po(3)+0.5*h)]];
Vertices(17:32,:)= [[Ci;Ci],[ones(8,1)*(po(3)-0.5*h);ones(8,1)*(po(3)-1.5*h)]];
b = ([-1 -1 -1 1 1 1]*(0.5*w)+po(3))';
Vertices(33:38,:)= [Vertices([8,1,2,10,9,16],1:2),b];
Vertices(39:42,:)= [Vertices([8,2,10,16],1:2),b(2:5)];
Vertices(39:42,1)= Vertices(39:42,1)+0.5*l;
p1 = po+[ro+0.5*l,0,0]; pt = p1+[0.5*l-ro,0,0];  p = [p1;pt];
Vertices(43,:) = p1;
Vertices(44,:) = pt;
probe=line(ax);set(probe,'xdata',Vertices(end-1:end,1),...
    'ydata',Vertices(end-1:end,2),...
    'zdata',Vertices(end-1:end,3),...
    "MarkerFaceColor",'red','Markersize',5,'Marker','s')

%% Genrate Faces
Faces = zeros(12,11); Faces(:,:) = nan;
Faces(1,1:11) = [1:5,21:-1:17,1];
Faces(2,1:11) = [5:8,1,17,24:-1:21,5];
Faces(3,1:9) = [25:32,25];
Faces(4,1:11) = [1:5,13:-1:9,1];
Faces(5,1:11) = [1,8:-1:5,13:16,9,1];
Faces(6,1:11) = Faces(4,1:11)+16;
Faces(7,1:11) = Faces(5,1:11)+16;
Faces(7,1:9) = Faces(3,1:9)-16;
Faces(8,1:6) = [36:38,42,41,36];
Faces(9,1:6) = [33:35,40,39,33];
Faces(10,1:5)= [35 36 41 40 35];
Faces(11,1:5)= [33 38 42 39 33];
Faces(12,1:5)= [39:42,39];
Link3.body=patch(ax,"Faces",Faces,'Vertices',Vertices,'FaceColor','green');
Link3.prob=probe;
end
```

[5] Kinematics

[5.1] 정기구학식 (Forward)

```
function point_data=Forward(linkdata,jointdata)
Np = size(jointdata,1);
L1 = linkdata(1); L2=linkdata(2); L3=linkdata(3);
point_data = zeros(Np,3);
for k = 1:Np
    alpha = sum(jointdata(k,2:3));
    point_data(k,1) = L2*cos(jointdata(k,2)) + L3*cos(alpha);
    point_data(k,2) = L2*sin(jointdata(k,2)) + L3*sin(alpha);
    point_data(k,3) = L1+jointdata(k,1);
end
end
```

[5.2] 역기구학식

```
function joint_data = Invese(link_data,point_data,sol)
%% function information
% get Joint angle -> invesekinematics
% input data  : link data  row 3 and col 1 [L1;L2;L3];
%             : point data row 3 and col 1 [px;py;pz];
%             : sol   -1 -> theta3 is pi< theta3 <2pi
%             : sol    1 -> theta3 is 0<=theta3<=pi
% output data:: joint data row 3 and col 1 [theta1; theta2; theta3];

joint_data = zeros(3,1);
x= point_data(1); y=point_data(2); z=point_data(3);
L1=link_data(1); L2 = link_data(2); L3 = link_data(3);
joint_data(1) = z - L1;
joint_data(3) = acos((x^2+y^2-L2^2-L3^2)/(2*L2*L3));

if sol == -1
    joint_data(3) =  -joint_data(3);
end

% Check
if ~isreal(joint_data(3))
    fprintf("warnning: theta3 is no real %f \n",joint_data(3));
    fprintf("Check this number x^{2}+y^{2}-L_{2}^{2}-L_{3}^{2}>=0: %f\n"...
        ,x^2+y^2-L2^2-L3^2);
    joint_data=[];
    return
end

k1 = L2+L3*cos(joint_data(3));
k2 = L3*sin(joint_data(3));
joint_data(2) = atan2(-k2*x+k1*y,k1*x+k2*y);
end
```

[5.3] 역기구학식에서 관절각 데이터 계산

```
function joint_data = get_joint(linkdata,pathcurve,sol)
Np = size(pathcurve,1);
joint_data = zeros(Np,3);
for k = 1:Np
    joint_data(k,:) = Invese(linkdata,pathcurve(k,:),sol);
```

```
    JM = Jacobian(linkdata,joint_data(k,:));
    if k ~= 1
        ND = det(JM([1,2],[2,3]));
        if PD * ND <=0
            fprintf("theta3 is zero so Changing Parameter\n")
        end
        PD = ND;
    else
        PD = det(JM([1,2],[2,3]));
    end
end
```

[5.4] 자코비안행렬

```
function JM = Jacobian(link_data,joint_data)
%% function information
% get Jacobian Matrix function:
% Jacobian Matrix size : row 6 and col 3
% input data : link data : row 1 and col 3 or row 3 and col 1
%           : joint_data: row 1 and col 3 or row 1 and col 3
% output data: JM -> Jacobian Matrix
%% function start
JM = [];
if isempty(joint_data)
    fprintf("t_v(Joint Angle data) is empty so, Check Joint Anlge \n");
    return
else
    [n,m]=size(joint_data);
    if n*m~=3
        fprintf("t_v(Joint Angle data) is 1x3 or 3x1 so, Check Link Length data
\n");
        return
    end
end

if isempty(link_data)
    fprintf("Link length vector is emtpy so, Check Link Length data \n");
    return
else
    [n,m]=size(link_data);
    if n*m == 3
        JM = zeros(6,3);
        JM([3,6],:) = [1 0 0;0 1 1];
        alp = sum(joint_data(2:3)); th2 = joint_data(2);
        JM(1:2,2:3)=[-link_data(3)*sin(alp);link_data(3)*cos(alp)]*[1,1]...
            + [-link_data(2)*sin(th2);link_data(2)*cos(th2)]*[0,1];
    else
        fprintf("Link length vector is 1x3 or 3x1 so, Check Link Length data \n");
    end
end
```

[6] Calculate(계산 function 파일)

[6.1] 강체변환행렬 계산코드

```
function G = get_G(u,q,t,h)
%GET_G 강체변환행렬을 생성
%------------information------------
%      여기서 u 는 회전축
```

```
%      여기서 q는 축상 임의 한점
%      t는 회전변환
%      h는 이동변환
%      로드리게스 회전변환 공식을 이용

us = [0 -u(3) u(2);...
    u(3) 0 -u(1);...
    -u(2) u(1) 0];

W = [u(1)^2-1 u(1)*u(2) u(1)*u(3);...
    u(2)*u(3) u(2)^2-1 u(2)*u(3);...
    u(1)*u(3) u(2)*u(3) u(3)^2-1];

I = eye(3);

R = I+(1-cos(t))*W+sin(t)*us;
c = (I-R)*q+h*u;
G = eye(4); G(1:3,:) = [R,c];
End
```

[6.2] 그림축의 축범위 계산

```
function axbox = get_box(xyz)

n = size(xyz,2);
min_data = min(xyz);
max_data = max(xyz);

axbox = reshape([min_data;max_data],1,n*2);
```

[6.3] 표준분포오차 matrix

```
function err = get_Error(sigma,mu,n,option)

err = normrnd(mu,sigma,[n,3]);

% varagin is add plotting option
if nargin >= 4
    if option == "on"
        f=figure(100); hold on; box on; pbaspect([1,1,1]); grid on;
        axis([mu-3*sigma,mu+3*sigma,0,200]);
        title("Error(Normal Distridution)");
        xlabel("Error[m]"); ylabel("Number")
        a = histfit(reshape(err,1,n*3));
    elseif option == "off"
        fprintf("Error Normar Distrubution ploting option off\n")
    else
        fprintf("option Error -> input value on or off \n")
    end
end
```

[6.4] 최소자승법계산식

```
function eps = Least_square(linkdata,jointdata,err)

Np = size(jointdata,1);
A1 = eye(3); A2 = zeros(3);
A  = zeros(Np*3,3);
b  = zeros(Np*3,1);
L1 = linkdata(1); L2 = linkdata(2); L3=linkdata(3);

for k = 1:Np
    A2(2,3) = cos(jointdata(k,3));
    A2(3,2) = cos(jointdata(k,3));
    alpha = sum(jointdata(k,2:3));

    A(3*(k-1)+1:3*k,1:3) = A1 + A2;

    b(3*(k-1)+1:3*k) = [err(k,3) - jointdata(k,1)-L1;...
        err(k,1)*cos(jointdata(k,2))+err(k,2)*sin(jointdata(k,2))-L2-
L3*cos(jointdata(k,3));...
        err(k,1)*cos(alpha)+err(k,2)*sin(alpha)-L2*cos(jointdata(k,3))-L3];
end

eps = inv(A'*A)*A'*b;
end
```

[6] Figure 관련 코드 -> pass by reference

[6.1] 축 범위 및 그래프 축 object 생성

```
function ax = mysetax(f1,title,lm,label,view)
% data : [Title, axis limit, grid on/off, label]
ax = axes();
ax.Parent= f1;
ax.Title = text(0.5,0.5,title);
ax.XLim = lm(1:2);
ax.YLim = lm(3:4);
ax.ZLim = lm(5:6);
ax.XLabel=text(0.5,0.5,label(1));
ax.YLabel=text(0.5,0.5,label(2));
ax.ZLabel=text(0.5,0.5,label(3));
ax.DataAspectRatio = [1,1,1];
ax.XGrid = 'on';ax.YGrid = 'on';ax.ZGrid = 'on';
ax.Box = 'on';
ax.XLimMode = 'manual';ax.YLimMode = 'manual';ax.ZLimMode='manual';
ax.View = view;
end
```

[6.2] 로봇팔의 애니메이션 코드

```
function moving_robot(ax,PRR,jointdata)
u = [0,0,1]';
Nj = size(jointdata,1);
Link2_Vertices = PRR.Link2.body.Vertices;
Link3_Vertices = PRR.Link3.body.Vertices;
pt_line = line(ax,'LineStyle','-','Color','black','Marker','*',...
        'Markersize',0.5);
pt_data = [];
ax.Title = text(20,20,"Moving Robot and Draw Path");

for k = 1:10:Nj
    G1 = get_G(u,PRR.origion',0,jointdata(k,1));
    G2 = get_G(u,PRR.p1',jointdata(k,2),0);
    G3 = get_G(u,PRR.p2',jointdata(k,3),0);

    % Link2 transform
    for k2 = 1:size(Link2_Vertices,1)
        dummy2 = G1*G2*[Link2_Vertices(k2,:)';1];
        PRR.Link2.body.Vertices(k2,:) = dummy2(1:3)';
    end
    % Link3 transform
    for k3 = 1:size(Link3_Vertices,1)
        dummy3 = G1*G2*G3*[Link3_Vertices(k3,:)';1];
        PRR.Link3.body.Vertices(k3,:) = dummy3(1:3)';
    end
    pt_data(end+1,:) = PRR.Link3.body.Vertices(end,:);
    set(pt_line,'xdata',pt_data(:,1),'ydata',pt_data(:,2),'zdata'...
        ,pt_data(:,3));
    set(PRR.Link3.prob,'xdata',PRR.Link3.body.Vertices(end-1:end,1),...
        'ydata',PRR.Link3.body.Vertices(end-1:end,2),...
        'zdata',PRR.Link3.body.Vertices(end-1:end,3));
    drawnow();
end
```

[6.3] 로봇의 작업공간 플로팅

```
function sca=draw_workspace(ax,PRR)

ax.View = [30,30];
ax.Subtitle = text(20,20,'WorkSpace');

u = [0,0,1]';
pt = [PRR.Link3.body.Vertices(end,:),1]';
N = 20;

th1 = linspace(0,5,N); % Height limit 0~5m;
th2 = linspace(0,2*pi,N);
th3  = th2;

dummypoint4 = zeros(4,1);
scatpoint = zeros(N*N*N,3);
index =1; temp=[];

for k1 =1:N
    for k2 =1:N
        for k3 = 1:N

        G1 = get_G(u,PRR.origion',0,th1(k1));
        G2 = get_G(u,PRR.p1',th2(k2),0);
        G3 = get_G(u,PRR.p2',th3(k3),0);

        dummypoint4(:) = G1*G2*G3*pt;
        scatpoint(index,:) = dummypoint4(1:3);
        index = index+1;

        end
    end
end

sca=scatter3(ax,scatpoint(:,1),scatpoint(:,2),scatpoint(:,3),'b*');
end
```

[6.4] 조인트각 비교 플로팅 코드

```
function compare_joint(fig,Jointdata,calibrationJointdata)

Xdata = 1:size(Jointdata,1);

ax1 = axes(fig); ax2=axes(fig); ax3=axes(fig);
ax1.XLabel = text(20,20,'number of data');
ax2.XLabel = text(20,20,'number of data');
ax3.XLabel = text(20,20,'number of data');

ax1.YLabel = text(20,20,'\theta_{1} [m]');
ax2.YLabel = text(20,20,'\theta_{2} [rad]');
ax3.YLabel = text(20,20,'\theta_{3} [rad]');

ax1.Title=text(20,20,'Comparing Change Joint Angle \theta_{1}');
ax2.Title=text(20,20,'Comparing Change Joint Angle \theta_{2}');
ax3.Title=text(20,20,'Comparing Change Joint Angle \theta_{3}');

ax1.XGrid="on"; ax1.YGrid="on";
```

```
ax2.XGrid="on"; ax2.YGrid="on";
ax3.XGrid="on"; ax3.YGrid="on";

t1 = line(ax1); c1 = line(ax1);
t2 = line(ax2); c2 = line(ax2);
t3 = line(ax3); c3 = line(ax3);

subplot(3,1,1,ax1)
legend(ax1,[t1,c1],["befor \theta_{1}","after \theta_{1}"])
subplot(3,1,2,ax2)
legend(ax2,[t2,c2],["befor \theta_{2}","after \theta_{2}"]);
subplot(3,1,3,ax3)
legend(ax3,[t3,c3],["befor \theta_{3}","after \theta_{3}"]);

set(t1,'xdata',Xdata,'ydata',Jointdata(:,1),'color','red');
set(t2,'xdata',Xdata,'ydata',Jointdata(:,2),'color','red');
set(t3,'xdata',Xdata,'ydata',Jointdata(:,3),'color','red');
set(c1,'xdata',Xdata,'ydata',calibrationJointdata(:,1),'color','blue');
set(c2,'xdata',Xdata,'ydata',calibrationJointdata(:,2),'color','blue');
set(c3,'xdata',Xdata,'ydata',calibrationJointdata(:,3),'color','blue');

legend()
```

[6.5] 링크의 길이 보정 전 후 비교 그래프 생성 코드

```
function compare_curve(fig,jointdata,linkdata,calibrationlinkdata)

point_data1=Forward(linkdata,jointdata);
point_data2=Forward(calibrationlinkdata,jointdata);
lb=get_box([point_data1;point_data2]);
ax1 = mysetax(fig,"Compare curve",lb,["X[m]","Y[m]","Z[m]"],[0,90]);
ax2 = mysetax(fig,"Compare curve",lb,["X[m]","Y[m]","Z[m]"],[30,45]);
path_line1 = line(ax1); path_line2 = line(ax2);
path_line3 = line(ax1); path_line4 = line(ax2);
subplot(1,2,1,ax1); subplot(1,2,2,ax2);
set(path_line1,'xdata',point_data1(:,1),'ydata',point_data1(:,2),...
    'zdata',point_data1(:,3),'Linestyle','-','color','red',...
    'linewidth',2);
set(path_line2,'xdata',point_data2(:,1),'ydata',point_data2(:,2),...
    'zdata',point_data2(:,3),'Linestyle','--','color','blue',...
    'linewidth',2);
legend(path_line1,'Robot Arm Path')
set(path_line4,'xdata',point_data1(:,1),'ydata',point_data1(:,2),...
    'zdata',point_data1(:,3),'Linestyle','-','color','red',...
    'linewidth',2);
set(path_line3,'xdata',point_data2(:,1),'ydata',point_data2(:,2),...
    'zdata',point_data2(:,3),'Linestyle','--','color','blue',...
    'linewidth',2);
legend([path_line1,path_line2],'Robot Arm Path','Calibration Path');

end
```

[6.6] 경로 곡선을 특정 axes 에 추가 코드

```
function pl=add_pathcurve(ax,pathcurve)
%% function information (animation view)
% input ax        : draw space (vareiable)
%       linkdata  : [L1,L2,L3] -> pt = [L1,L2,L3];
%       jointdata : [t1,t2,t3]
%       pathcurve : [x,y,z]
%       option    : defualt "off", "on" The link of the robot is simply
%                   expressed with a line.
%% =========================initial setting================================
Np= size(pathcurve,1);
Pl=line(ax);
set(Pl,'xdata',pathcurve(:,1),'ydata',pathcurve(:,2),...
    'zdata',pathcurve(:,3),'Linestyle','--','Linewidth',1,'Color','red');
End
```

[6.7] 특정 axes 에 오차경로 곡선 추가 코드

```
function errpathcurve=add_ecurve(ax,errpathdata)

Np = size(errpathdata,1);
errpathcurve = line(ax);
errpathcurve.XData = [errpathdata(:,1);errpathdata(1,1)];
errpathcurve.YData = [errpathdata(:,2);errpathdata(1,2)];
errpathcurve.ZData = [errpathdata(:,3);errpathdata(1,3)];
set(errpathcurve,'LineStyle','none','Marker','s',...
    'MarkerEdgeColor','r','Markersize',3)
end
```