1 Homework Assignment (HA)

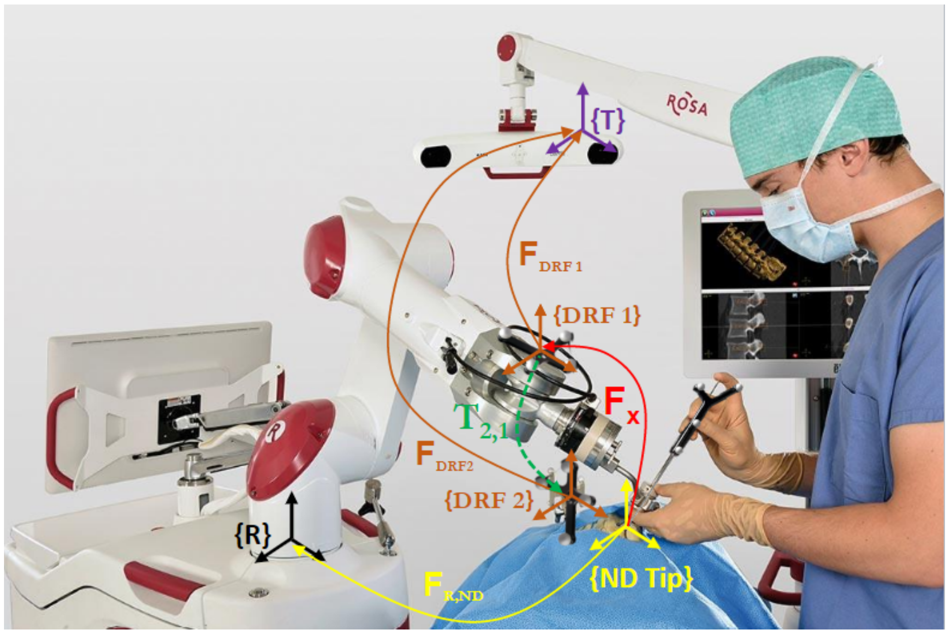


Figure 1: The Rosa Spine Surgical system

The patient's anatomy has been already registered to the pre-op CT using only the navigation system.

1.1 Given

Transformation between the Dynamic Reference Frame (DRF) and the sensor frame : $F_{T,DRF1}, F_{T,DRF2}$; The transformation between the tip of the surgical navigation device (ND Tip) and the robot base:$F_{R,ND}$

1.2 Unknowns

The transformation between the sensor frame and the world frame : $F_{T,R}$. The transformation between the DRF1 and the tip of the ND: $F_{DRF1,ND}$.

1.3 Eye to Hand Calibration (AY=YB) Algorithm

We can pick up two loops to form equations.

$$F_{T,ND} = F_{T,R}F_{R,ND} = F_{T,DRF1}F_{DRF1,ND}$$

To simply the representation of the equations, we can use Z to represent $F_{T,R}$, and use X to represent$F_{DRF1,ND}$. Similarly, we use S to represent $F_{R,ND}$ and use E to represent $F_{T,DRF1}$.

Then the equation becomes

$$ZS = EX$$

- Step1: move the robot to an arbitrary configuration E_1.
- Step2: measure the configuration of the DRF using sensor S_1.
- Step3: move the robot to an arbitrary configuration E_k for $k = 1, 2, ...N$.
- Step4: measure the configuration of the DRF using sensor S_k for $k = 1, 2, ...N$.

$$ZS_1 = E_1X$$

From this formula, we can get

$$Z = E_1XS_1^{-1}$$

$$ZS_k = E_kX$$

$$E_1XS_1^{-1}S_k = E_kX$$

$$XS_1^{-1}S_k = E_1^{-1}E_kX$$

$$A_kX = XB_k$$

A_k represents motion from an initial pose $E_1A_k = E_k$.

Similarly,B_k represents motion from an initial pose $S_1B_k = S_k$.

1.4 Solution of the (AX=XB) Algorithm

We will use axis-angle to solve this equation. We can first solve for the rotation, then solve for the translation.

$$\begin{pmatrix} R_A & t_A \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R_X & t_X \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_B & t_B \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R_X & t_X \\ 0 & 1 \end{pmatrix}$$

Step1: $R_AR_X = R_BR_X$ Solve the rotation matrix R_x with more than two data points: If we have $A_1, ..., A_N, B_1, ..., B_N$, then we want to R_x and t_x to 'best fit the data'.

We start again by first solving for the rotation matrix and define matrix M as the following:

$$M = \sum_{i=1}^{N} \beta_i\alpha_i^T$$

$$\alpha_i = log(R_{Ai}), \beta_i = log(R_{Bi})$$

Using the polar decomposition, it is provided that:

$$R_{DRF1,ND} = R_x = (M^TM)^{-1/2}M^T$$

Using Eigen decomposition we have: $(M^TM)^{-1/2} = Q\Delta^{-1/2}Q^{-1}$

Step2: $R_At_X + t_A = t_BR_X + t_X$ Solving for the translation vector with more than two data points: Once R_x is found, we solve for t_x using least squares:

$$\begin{pmatrix} I - R_{A_1} \\ \vdots \\ I - R_{A_N} \end{pmatrix} t_X = \begin{pmatrix} t_{A_1} - t_{B_1} R_X \\ \vdots \\ t_{A_N} - R_X t_{B_N} \end{pmatrix}$$

$$t_{DRF1,ND} = t_x$$

$$F_{DRF1,ND} = \begin{pmatrix} R_{DRF1,ND} & t_{DRF1,ND} \\ 0 & 1 \end{pmatrix}$$

2 Programming Assignment (PA1)

2.1 Develop a 3D point set to 3D point set registration algorithm

Develop a 3D point set to 3D point set registration algorithm. The output of a point set registration algorithm is the optimal transformation T^* such that M is best aligned to S according to some defined notion of distance function dist(...):

$$T^* = argmin_{T \in \tau} dist(T(M), S)$$

Then taking the square of the Euclidean distance for every pair of points:
finding corresponding points:

$$s_m = argmin_{s \in S} \parallel s - m \parallel_2^2$$

finding mapping:

$$dist(T(M), S) = \sum_{m \in T(M)} \parallel m - s_m \parallel_2^2$$

When the correspondences are given before the optimization, the registration is correspondence-based registration. The points $s_i and m_i \in R^3$ are generated as follows:

$$s_i = lRm_i + t + \epsilon_i, i = 1, ..., N$$

R is rotation matrix
t is translation vector.
Given: point cloud(one measurement source, one rigid object and two different configurations)
Find: the transformation $T_{bb'} = [R, p]$ between these two configurations.
Step1: compute $\bar{a}$ and $\bar{b}$

$$\bar{a} = \frac{1}{N} \sum_{i=1}^{N} \bar{a_i}$$

$$\bar{b} = \frac{1}{N} \sum_{i=1}^{N} \bar{b_i}$$

Step2: compute $\tilde{a}$ and $\tilde{b}$

$$\tilde{a_i} = \bar{a_i} - \bar{a}$$

$$\tilde{b}_i = \bar{b_i} - \bar{b}$$

Step3: find R that minimizes:

$$\sum_i (R\tilde{a}_i - \tilde{b}_i)^2$$

Step4: Find p

$$\bar{p} = \bar{b} - R\bar{a}$$

Step5: Desired transformation is: $T_{\bar{a}\bar{b}} = [R, p]$ for step 3, we can find R using the SVD approach:
step 3.1 computer matrix H:

$$H = \sum_i \begin{pmatrix} \tilde{a}_{i,x}\tilde{b}_{i,x} & \tilde{a}_{i,x}\tilde{b}_{i,y} & \tilde{a}_{i,x}\tilde{b}_{i,z} \\ \tilde{a}_{i,y}\tilde{b}_{i,x} & \tilde{a}_{i,y}\tilde{b}_{i,y} & \tilde{a}_{i,y}\tilde{b}_{i,z} \\ \tilde{a}_{i,z}\tilde{b}_{i,x} & \tilde{a}_{i,z}\tilde{b}_{i,y} & \tilde{a}_{i,z}\tilde{b}_{i,z} \end{pmatrix}$$

step 3.2: computer SVD of H:

$$H = USV^T$$

step 3.3: $R = VU^T$
step 3.4: verify $det(R) = 1$

2.2 Develop a "pivot" calibration method

Pivot calibration is used to find the position of rigid drill bit $\bar{b_{tip}}$ with respect to a DRF.

Given a set of known rigid transformations $F^k_{T,pointer} = F_k$ obtained by pivoting a pointer tool around a fixed world point, estimate the unknown translation $\bar{b_{tip}}$ from the dynamic reference frame(DRF) origin to the pivoting point and the unknown translation $\bar{b_{post}}$ from the tracker origin to the pivoting point.

$$\bar{b_{post}} = F_k \bar{b_{tip}} = R_k \bar{b_{tip}} + \bar{p_k}$$

From each measurement k, we have known R_k and p_k:

$$\bar{b_{post}} = R_k \bar{b_{tip}} + \bar{p_k}$$

we can rewrite this equation as:

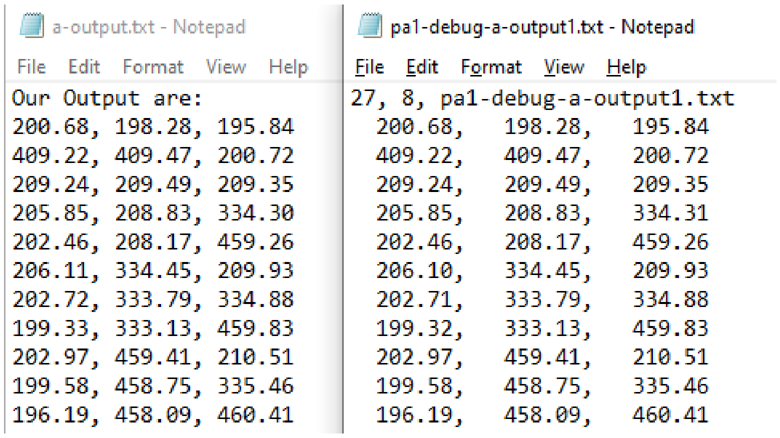
$$R_k \bar{b_{tip}} - \bar{b_{post}} = -\bar{p_k}$$

So we can set up a least square problem in the form of $Ax = b$ as follows and find the unknowns $\bar{b_{tip}}$ and $\bar{b_{post}}$

$$\begin{pmatrix} \vdots & \vdots \\ R_k & -I \\ \vdots & \vdots \end{pmatrix} \begin{pmatrix} \bar{b}_{\text{tip}} \\ \bar{b}_{\text{post}} \end{pmatrix} = \begin{pmatrix} \vdots \\ -\bar{p_k} \\ \vdots \end{pmatrix}$$

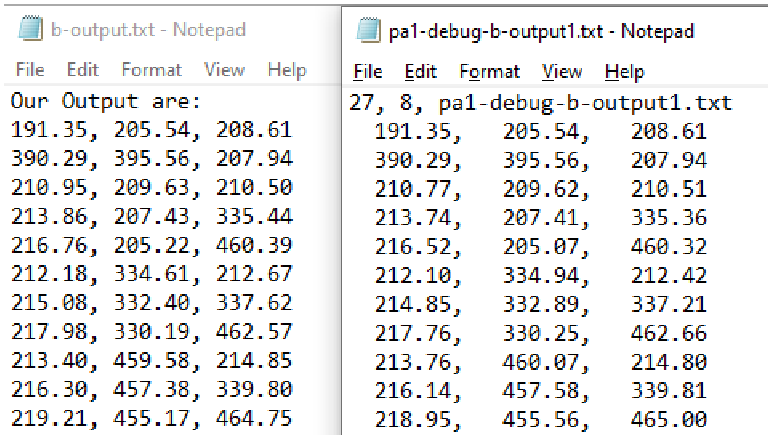
2.3 PA1: goals: 3,4,5 output

For debug data (a-g), the comparison are shown as follows(left is our Output, right is provided output):


a-output.txt - Notepad
File Edit Format View Help
pa1-debug-a-output1.txt - Notepad
File Edit Format View Help

```
Our Output are:
200.68, 198.28, 195.84
409.22, 409.47, 200.72
209.24, 209.49, 209.35
205.85, 208.83, 334.30
202.46, 208.17, 459.26
206.11, 334.45, 209.93
202.72, 333.79, 334.88
199.33, 333.13, 459.83
202.97, 459.41, 210.51
199.58, 458.75, 335.46
196.19, 458.09, 460.41
```

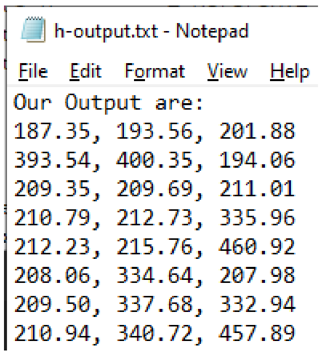
```
27, 8, pa1-debug-a-output1.txt
  200.68,    198.28,    195.84
  409.22,    409.47,    200.72
  209.24,    209.49,    209.35
  205.85,    208.83,    334.31
  202.46,    208.17,    459.26
  206.10,    334.45,    209.93
  202.71,    333.79,    334.88
  199.32,    333.13,    459.83
  202.97,    459.41,    210.51
  199.58,    458.75,    335.46
  196.19,    458.09,    460.41
```


b-output.txt - Notepad
File Edit Format View Help
pa1-debug-b-output1.txt - Notepad
File Edit Format View Help

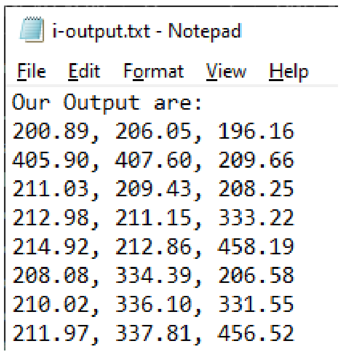
```
Our Output are:
191.35, 205.54, 208.61
390.29, 395.56, 207.94
210.95, 209.63, 210.50
213.86, 207.43, 335.44
216.76, 205.22, 460.39
212.18, 334.61, 212.67
215.08, 332.40, 337.62
217.98, 330.19, 462.57
213.40, 459.58, 214.85
216.30, 457.38, 339.80
219.21, 455.17, 464.75
```

```
27, 8, pa1-debug-b-output1.txt
  191.35,    205.54,    208.61
  390.29,    395.56,    207.94
  210.77,    209.62,    210.51
  213.74,    207.41,    335.36
  216.52,    205.07,    460.32
  212.10,    334.94,    212.42
  214.85,    332.89,    337.21
  217.76,    330.25,    462.66
  213.76,    460.07,    214.80
  216.14,    457.58,    339.81
  218.95,    455.56,    465.00
```

For unknown data(h-k), our output are shown here:



```
h-output.txt - Notepad
Our Output are:
187.35, 193.56, 201.88
393.54, 400.35, 194.06
209.35, 209.69, 211.01
210.79, 212.73, 335.96
212.23, 215.76, 460.92
208.06, 334.64, 207.98
209.50, 337.68, 332.94
210.94, 340.72, 457.89
```



```
i-output.txt - Notepad
Our Output are:
200.89, 206.05, 196.16
405.90, 407.60, 209.66
211.03, 209.43, 208.25
212.98, 211.15, 333.22
214.92, 212.86, 458.19
208.08, 334.39, 206.58
210.02, 336.10, 331.55
211.97, 337.81, 456.52
```

All the data Output(a-k) are in our folder, names are 'x-output.txt'. Here we just showed parts of our result.

3 Programming Assignment (PA2)

For eye in hand algorithm: E1 and E2 are given by the forward kinematics.

S1 and S2 are given by measuring the position of the object wrt to the sensor.

X is the unknown transformation between the sensor and robot end-effector frames.

$$E_1XS1 = E_2XS_2$$
$$XS_1S_2^{-1} = E_1^{-1}E_2X$$
$$AX = XB$$

Note: A and B are defined based on two consecutive configurations of the robot and sensor (i.e. configuration 1 and configuration 2)

3.1 Non-Noise Data

3.1.1 Axis-Angle analytical approach

We will use axis-angle to solve this equation. We can first solve for the rotation, then solve for the translation.

$$\begin{pmatrix} R_A & t_A \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R_X & t_X \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} R_B & t_B \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R_X & t_X \\ 0 & 1 \end{pmatrix}$$

-

Step1: $R_AR_X = R_BR_X$ Solve the rotation matrix R_x with more than two data points: If we have $A_1, ..., A_N, B_1, ..., B_N$, then we want to R_x and t_x to 'best fit the data'.

We start again by first solving for the rotation matrix and define matrix M as the following:

$$M = \sum_{i=1}^{N} \beta_i \alpha_i^T$$

$$\alpha_1 = log(R_{A1}), \beta_1 = log(R_{B1})$$

Using the polar decomposition, it is provided that:

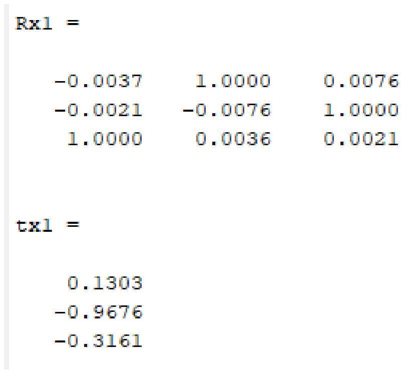
$$R_x = (M^T M)^{-1/2} M^T$$

Using Eigen decomposition we have: $(M^T M)^{-1/2} = Q\Delta^{-1/2}Q^{-1}$

Step2: $R_A t_X + t_A = t_B R_X + t_X$ Solving for the translation vector with more than two data points: Once R_x is found, we solve for t_x using least squares:

$$\begin{pmatrix} I - R_{A_1} \\ \vdots \\ I - R_{A_N} \end{pmatrix} t_X = \begin{pmatrix} t_{A_1} - t_{B_1} R_X \\ \vdots \\ t_{A_N} - R_X t_{B_N} \end{pmatrix}$$

The result for this part is as follows:



```
Rx1 =

   -0.0037    1.0000    0.0076
   -0.0021   -0.0076    1.0000
    1.0000    0.0036    0.0021


tx1 =

    0.1303
   -0.9676
   -0.3161
```

3.1.2 Quaternion analytical approach

The similar procedure as above, we need to get q_A and q_B. Using the function we wrote in the previous programming assignment, we can easily calculate the quaternion product from the two different configurations.

$$q_A = q_{E_1}^{-1} \circ q_{E_2}, q_B = q_{S_1} \circ q_{S_2}^{-1}$$

We split the quaternion into the scalar and vector parts, i.e. $q_x = s_x + \vec{v_x}$. Then we can find R_x and p_x:

- Step 1: Find R_x
  For $R_A R_X = R_X R_B$, we have $q_A \circ q_x = q_x \circ q_B$, after rearranging, we get:

$$(s_A - s_B)s_x - (\vec{v_A} - \vec{v_B}) \cdot \vec{v_X} = 0$$

$$(\vec{v_A} - \vec{v_B})s_X + (s_A - s_B)\vec{v_X} + (\vec{v_A} + \vec{v_B}) \times \vec{v_X} = \vec{0}$$

  Expressing this as an $Ax = b$ matrix equation:

$$Mq = \begin{bmatrix} (s_A - s_B) & (\vec{v_A} - \vec{v_B})^T \\ (\vec{v_A} - \vec{v_B}) & (s_A - s_B)I_3 + skew(\vec{v_A} + \vec{v_B}) \end{bmatrix} \begin{bmatrix} s_X \\ \vec{v_X} \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{0} \end{bmatrix}$$

For this question, we generated more than two data points, thus we can get the M matrix as:

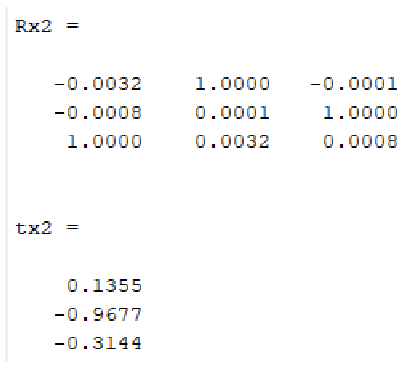
$$\begin{bmatrix} M(q_{A_1}, q_{B_1}) \\ M(q_{A_2}, q_{B_2}) \\ \dots \\ M(q_{A_N}, q_{B_N}) \end{bmatrix}$$

By using the Singular Value Decomposition ($[U, S, V] = svd(M)$), we can get $4th$ column of V and the unit quaternion representation of R_X.

- Step 2: Find p_X
  The sample approach as before, once R_X is found, we solve for t_x using least squares:

$$\begin{pmatrix} I - R_{A_1} \\ \vdots \\ I - R_{A_N} \end{pmatrix} t_X = \begin{pmatrix} t_{A_1} - t_{B_1} R_X \\ \vdots \\ t_{A_N} - R_X t_{B_N} \end{pmatrix}$$

The result for this part is as follows:



```
Rx2 =

   -0.0032    1.0000   -0.0001
   -0.0008    0.0001    1.0000
    1.0000    0.0032    0.0008


tx2 =

    0.1355
   -0.9677
   -0.3144
```

3.1.3 Comparison

Compare the results using reasonable error measures for both rotational and translation parts. Here we use the norm of matrix and vector to present the error:

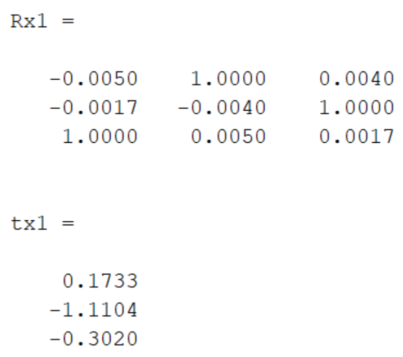
$$norm(R_{X_1} - R_{X_2}),\ norm(P_{X_1}, P_{X_2})$$

The error measure for rotational part is: 0.00782721.
The error measure for translational part is: 0.00546166

3.2 Noise Data

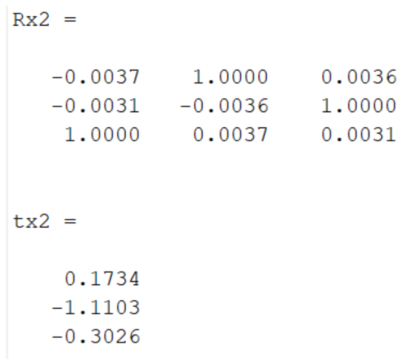
Using half of data sets, we get the result as follows:

3.2.1 Axis-Angle analytical approach



```
Rx1 =

   -0.0050    1.0000    0.0040
   -0.0017   -0.0040    1.0000
    1.0000    0.0050    0.0017


tx1 =

    0.1733
   -1.1104
   -0.3020
```

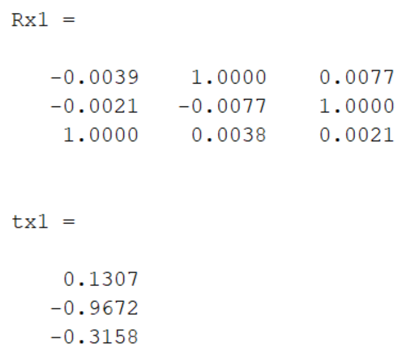
3.2.2 Quaternion analytical approach



```
Rx2 =

   -0.0037    1.0000    0.0036
   -0.0031   -0.0036    1.0000
    1.0000    0.0037    0.0031


tx2 =

    0.1734
   -1.1103
   -0.3026
```

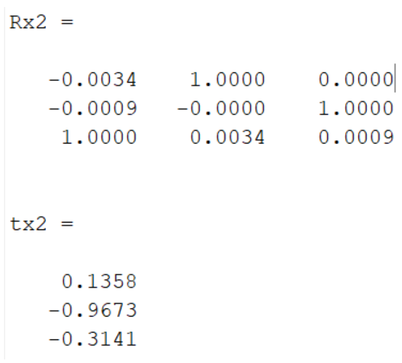
Using the whole data sets, we get the result as follows:

3.2.3 Axis-Angle analytical approach



```
Rx1 =

   -0.0039    1.0000    0.0077
   -0.0021   -0.0077    1.0000
    1.0000    0.0038    0.0021


tx1 =

    0.1307
   -0.9672
   -0.3158
```

3.2.4 Quaternion analytical approach

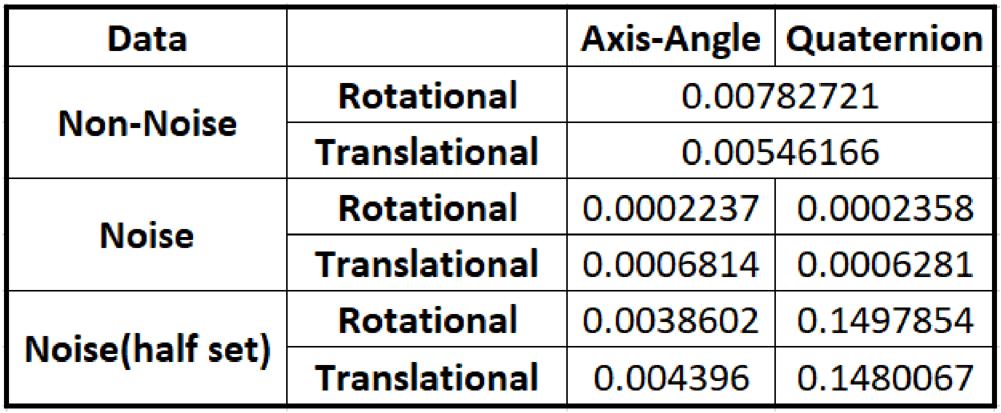


```
Rx2 =

   -0.0034    1.0000    0.0000
   -0.0009   -0.0000    1.0000
    1.0000    0.0034    0.0009


tx2 =

    0.1358
   -0.9673
   -0.3141
```

3.2.5 Compare the results and discuss the results

In order to make a comparison between the noise free data and noisy data, we calculate the norm (2-norm) of the difference between noisy data and noise free data.

When we use more data sets, the errors decrease and we can get more accurate results. So the results for the whole data sets are more accurate than that of half data sets.



Data		Axis-Angle	Quaternion
Non-Noise	Rotational	0.00782721	
	Translational	0.00546166	
Noise	Rotational	0.0002237	0.0002358
	Translational	0.0006814	0.0006281
Noise(half set)	Rotational	0.0038602	0.1497854
	Translational	0.004396	0.1480067

Figure 2: comparison of results between noise free data and noisy data