Overview

Monday, October 25, 2021 19:28

Name 1	HongYi Fan
Email	hfan15@jhu.edu
Other contact information (optional)	
Name 2	Yuxin Chen
Email	ychen506@jhu.edu
Other contact information (optional)	
Signature (required)	I (we) have followed the rules in completing this assignment _____ Yuxin Chen _____

Overview

Contents of this document:

- 3D point set to 3D point set registration algorithm and mathematical approach
- Program structure outline for 3D point set to 3D point set registration
- Algorithm of computation of the expected values $\overrightarrow{C_{i(expected)}}$ for the $\vec{C}$
- Results of computation of the expected values $\overrightarrow{C_{i(expected)}}$ for the $\vec{C}$
- Mathematical approach of the "Pivot" Calibration Algorithm
- Program structure outline for "Pivot" Calibration
- Results of "Pivot" Calibration

A short statement of who did what:

Yuxin Chen:

We went through the whole assignment together and discussed the algorithm and mathematical approach we decided to use. Then I mainly wrote the function and report for the 3D point to point registration and computing the C_expected value. We wrote the report together.

HongYi Fan:

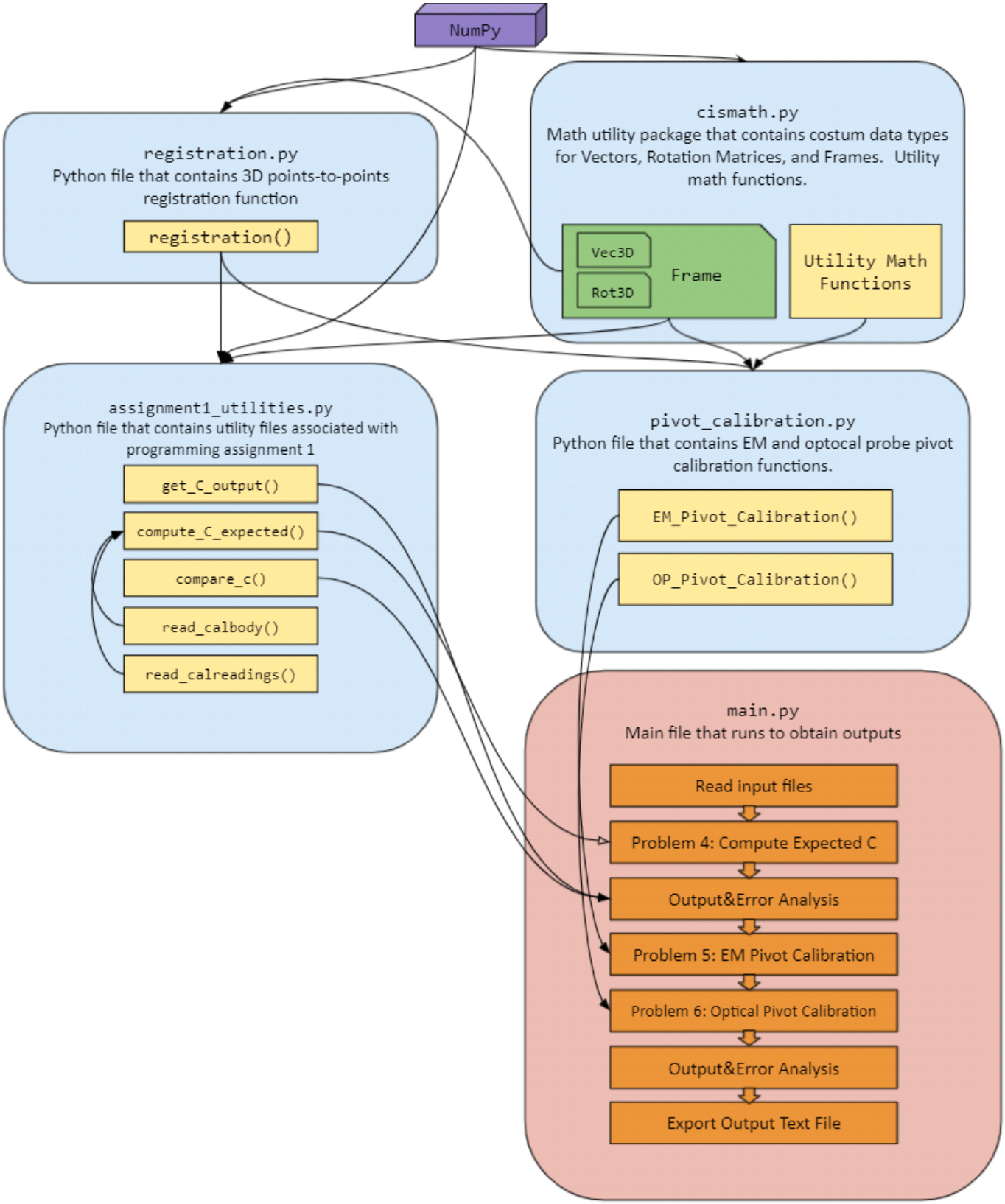
Went through the assignment and algorithms with teammate. Wrote the majority of the code for pivot calibration and cismath package.

External Resources:

- Numpy (https://numpy.org/)

Program Structure Overview

Tuesday, October 26, 2021 1:19 PM



The lines and arrows between boxes indicates involvement of an element in another.
For example, A → B means function B calls or involve function A.

Algorithm and Mathematical Approach for 3D point set to 3D point set registration

Monday, October 25, 2021 19:29

3D point set to 3D point set registration algorithm

Known:

- Input: two 3D point sets, $\vec{a}_i$ *and* $\vec{b}_i$

Goal:

- Output: Frame that transform first 3D point set $\vec{a}_i$ to second 3D point set $\vec{b}_i$

Steps :

1. Compute the average points of first sets of points. Then for each point in first sets of points, calculate the $\tilde{a}$ by using the following equation. Do the same procedure to second sets of points

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

$$\tilde{a}_i = \vec{a}_i - \bar{a}$$

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

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

1. Find R that minimizes

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

To find R, we used a quaternion method for R. Details are shown in the next mathematical approach section.

2. Find $\vec{p}$ by using the R we found in step 2 and average point of first and second sets of points we calculated in step 1.

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

4. Desired transform is

F = ***Frame(R,*** $\vec{p}$)

The mathematical approach taken

In the step 2 of a 3D point set to 3D point set registration algorithm, we used a quaternion method to find R that minimizes

$$\sum_i \left(\boldsymbol{R} \cdot \tilde{\boldsymbol{a}}_i - \tilde{\boldsymbol{b}}_i\right)^2$$

The details of this mathematical approach is shown below:

1. Let q = s + $\vec{v}$ be the unit quaternion corresponding to R.
   Let $\tilde{a}\ and\ \tilde{b}$ be the vectors with $\tilde{\boldsymbol{b}} = \boldsymbol{R} \cdot \tilde{\boldsymbol{a}}$
   Then we can have a quaternion equation
   Because $(\mathbf{s} - \vec{\boldsymbol{v}})(\mathbf{s} + \vec{\boldsymbol{v}}) = \mathbf{1} + \vec{\mathbf{0}}$
   $(\mathbf{s} + \vec{\boldsymbol{v}}) \cdot (\mathbf{0} + \tilde{\boldsymbol{a}})(\boldsymbol{s} - \vec{\boldsymbol{v}}) = \mathbf{0} + \tilde{\boldsymbol{b}}$
   $(\mathbf{s} + \vec{\boldsymbol{v}}) \cdot (\mathbf{0} + \tilde{\boldsymbol{a}}) = \left(\mathbf{0} + \tilde{\boldsymbol{b}}\right) \cdot (\mathbf{s} + \vec{\boldsymbol{v}})$

   Expanding the scalar and vector parts gives:
   $-\vec{\boldsymbol{v}} \cdot \tilde{\boldsymbol{a}} = -\vec{\boldsymbol{v}} \cdot \tilde{\boldsymbol{b}}$
   $\mathbf{s}\tilde{\boldsymbol{a}} + \vec{\boldsymbol{v}} \times \tilde{\boldsymbol{a}} = \mathbf{s}\tilde{\boldsymbol{b}} + \tilde{\boldsymbol{b}} \times \vec{\boldsymbol{v}}$
   Rearranging:
   $(\tilde{\boldsymbol{b}} - \tilde{\boldsymbol{a}}) \cdot \vec{\boldsymbol{v}} = \mathbf{0}$
   $\mathbf{s}(\tilde{\boldsymbol{b}} - \tilde{\boldsymbol{a}}) + (\tilde{\boldsymbol{b}} + \tilde{\boldsymbol{a}}) \times \vec{\boldsymbol{v}} = \vec{\mathbf{0}}$

2. Express the relation for each point $\vec{a}$ and corresponding $\vec{b}$ as a matrix equation:
   $$\begin{bmatrix} \mathbf{0} & (\tilde{\boldsymbol{b}} - \tilde{\boldsymbol{a}})^{\boldsymbol{T}} \\ (\tilde{\boldsymbol{b}} - \tilde{\boldsymbol{a}}) & \boldsymbol{sk}(\tilde{\boldsymbol{b}} + \tilde{\boldsymbol{a}}) \end{bmatrix} \begin{bmatrix} \boldsymbol{s} \\ \vec{\boldsymbol{v}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \overrightarrow{\mathbf{0}_3} \end{bmatrix}$$

   Express $\vec{\boldsymbol{q}} = \begin{bmatrix} \boldsymbol{s} \\ \vec{\boldsymbol{v}} \end{bmatrix}$

   $$\mathbf{M}(\vec{\boldsymbol{a}}, \vec{\boldsymbol{b}}) = \begin{bmatrix} \mathbf{0} & (\tilde{\boldsymbol{b}} - \tilde{\boldsymbol{a}})^{\boldsymbol{T}} \\ (\tilde{\boldsymbol{b}} - \tilde{\boldsymbol{a}}) & \boldsymbol{sk}(\tilde{\boldsymbol{b}} + \tilde{\boldsymbol{a}}) \end{bmatrix}$$

   Then $\mathbf{M}(\vec{\boldsymbol{a}}, \vec{\boldsymbol{b}})\ \vec{\boldsymbol{q}} = \overrightarrow{\mathbf{0}_4}$
   Where $\|\vec{\boldsymbol{q}}\| = \mathbf{1}, since\ it's\ unit\ quaternion$

3. We want to solve the problem in a least squares sense:
   $\min \|M\vec{q}\|\ subject\ to\ \|\vec{\boldsymbol{q}}\| = \mathbf{1}$
   Where
   $$\mathbf{M} = \begin{bmatrix} \mathbf{M}(\overrightarrow{\boldsymbol{a_1}}, \overrightarrow{\boldsymbol{b_1}}) \\ \vdots \\ \mathbf{M}(\overrightarrow{\boldsymbol{a_n}}, \overrightarrow{\boldsymbol{b_n}}) \end{bmatrix}$$ where n is the number of observations

   Taking the singular value decomposition of $\mathbf{M} = \mathbf{U}\sum \boldsymbol{V^T}$ reduces this to easier problem

   $$\mathbf{min}\left\|\mathbf{U}\sum \boldsymbol{V^T}\,\overrightarrow{\boldsymbol{q_x}}\right\| = \left\|\boldsymbol{U}(\sum \vec{\boldsymbol{y}})\right\| = \left\|\sum \vec{\boldsymbol{y}}\right\| \textbf{ subject to } \|\vec{\boldsymbol{y}}\| = \left\|\boldsymbol{V^T}\vec{\boldsymbol{q}}\right\| = \|\vec{\boldsymbol{q}}\| = \mathbf{1}$$

4. Then the problem is
   $$\mathbf{min}\left\|\sum \vec{\boldsymbol{y}}\right\| = \left\|\begin{bmatrix} \boldsymbol{\sigma_1} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{\sigma_4} \end{bmatrix} \vec{\boldsymbol{y}}\right\| \ \boldsymbol{subject\ to}\ \|\vec{\boldsymbol{y}}\| = \mathbf{1}$$
   Where σ_i are the singular valuses

   SVD routines typically return $\sigma_i \geq 0$ and sorted in decreading magnitude.

Thus σ_4 is the smallest singular value and the value of $\vec{y}$ with $\|\vec{y}\| = \mathbf{1}$ that minimizes $\|\sum \vec{y}\|$ **is** $\vec{y} = [\mathbf{0}, \mathbf{0}, \mathbf{0}, \mathbf{1}]^{T}$.
The corresponding value of $\vec{q}$ is given by $\vec{q} = V\vec{y} = V_4$. Where V_4 is the 4th column of V.

5. After we get $\vec{q}$, we calculate the corresponding R by using the following equation:
   Since $\vec{q}$ is the unit quaterion correspoind to R,
   $\vec{q} = [q_0, q_1, q_2, q_3]$

$$R = \begin{bmatrix} {q_0}^2 + {q_1}^2 - {q_2}^2 - {q_3}^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & {q_0}^2 - {q_1}^2 + {q_2}^2 - {q_3}^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & {q_0}^2 - {q_1}^2 - {q_2}^2 + {q_3}^2 \end{bmatrix}$$

Algorithm of computating of the expected values $\overrightarrow{C_{i(expected)}}$ for the $\overrightarrow{C}$

Tuesday, October 26, 2021 8:59 PM

Algorithm of computation of the expected values $\overrightarrow{C_{i(expected)}}$ for the $\vec{C}$

Known:

- $\mathrm{N_D}$: Number of optical markers on EM base
- $\mathrm{N_A}$: number of optical markers on calibration object
- N_C: number EM markers on calibration object
- N_{frames}: number of "data frames" of data
- Coordinates of $\overrightarrow{d_i}$
- Coordinates of $\overrightarrow{a_i}$
- Coordinates of $\overrightarrow{c_i}$
- Each calibration data frame $[\overrightarrow{D_1}, \cdots, \overrightarrow{D_{N_D}}, \overrightarrow{\mathrm{A}_1}, \cdots, \overrightarrow{A_{N_A}}, \overrightarrow{C_1}, \cdots, \overrightarrow{C_{N_D}}]$

Goal:

- Calculate the "expected" values $\overrightarrow{C_{i(expected)}}$ $for\ the\ \overrightarrow{C_i}$

STEPS:

1. For each calibration data frame $[\overrightarrow{D_1}, \cdots, \overrightarrow{D_{N_D}}, \overrightarrow{\mathrm{A}_1}, \cdots, \overrightarrow{A_{N_A}}, \overrightarrow{C_1}, \cdots, \overrightarrow{C_{N_D}}]$, calculate the F_D between optical tracker and EM tracker coordinates.

   Since $\overrightarrow{D_i} = F_D \cdot \overrightarrow{d_i}$,
   To find F_D, we used the 3D point set to 3D point set registration algorithm we developed in Question 2.
   We set coordinates of $\overrightarrow{d_i}$ as the first 3D point set and $\overrightarrow{D_i}$ as the second 3D point set. Then the algorithm will calculate F_D

2. Then calculate the F_A between calibration object and optical tracker coordinates.

   Since $\overrightarrow{A_i} = F_A \cdot \overrightarrow{a_i}$,
   To find F_A, we used the 3D point set to 3D point set registration algorithm we developed in Question 2.
   We set coordinates of $\overrightarrow{a_i}$ as the first 3D point set and $\overrightarrow{A_i}$ as the second 3D point set. Then the algorithm will calculate F_A

3. After we get $F_D\ and\ F_A$, we compute the $\overrightarrow{C_{i(expected)}}$ by using the following equation
   $\overrightarrow{C_{i(expected)}} = {F_A}^{-1} \cdot F_D \cdot \overrightarrow{c_i}$

4. Then we get the output: $\overrightarrow{C_{i(expected)}}$

Algorithm of "Pivot" Calibration

Monday, October 25, 2021 19:32

"Pivot" Calibration Algorithm

Known:

- Multiple data frames of "Positions of markers fixed on a tool, G_i, in respective to some fixed coordinate frame", during a pivot calibration procedure.
- Tool tip is at the some position P_{dimple} during the calibration process

Goal:

- Calculate $\vec{p}_t$ that represents the displacement from tool tip to the tool trackers (centroid of all tracker markers)

STEPS:

1. Compute a reference coordinate frame using the first frame of calibration data

$$G_0 = \frac{1}{\#of\ markers} \cdot \sum_{i}^{\#of\ markers} G_i$$

This position vector, G_0 is used as a reference position

$$\vec{g}_i = G_i - G_0$$

The set of vectors $g_i[i = 1 \dots \#of\ markers]$ describes the displacement between each marker and the centroid of the markers. These vectors are used to calculate the pose of the tool. Note the g_i is the same for all data frames.

2. For each data frame, compute the pose of the tool relative to the tracker base.

For the k_{th} data frame, the transformation F_k satisfies:
$F_k \cdot \vec{g}_i = \vec{G}_i^k$
Where $\vec{g}_i$ is the set of vector computed in step 1, G_i^k {i = 0… #of markers} is the set of locations of markers in the kth data frame.

F_k can be computed using the registration algorithm developed in part 2.
Now we have F_k that is the transformation between the initial data frame and the kth data frame

3. Compute p_t using a set of F_k

It is known that in the kth data frame:
$F_k \cdot \vec{p}_t = \vec{p}_{pivot}$
$R_k\vec{p}_t + \vec{p}_k = \vec{p}_{pivot}$
$R_k\vec{p}_t - \vec{p}_{pivot} = -\vec{p}_k$

We can write above equation in the form below:

$$\begin{bmatrix} \vdots & \vdots \\ R_k & -I \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} \vec{p}_t \\ \vec{p}_{pivot} \end{bmatrix} = \begin{bmatrix} \vdots \\ -\vec{p}_k \\ \vdots \end{bmatrix}$$

Since R_k, I, p_k are known values, we can use a least square root method to calculate $\begin{bmatrix} \vec{p}_t \\ \vec{p}_{pivot} \end{bmatrix}$

$$A = \begin{bmatrix} \vdots & \vdots \\ R_k & -I \\ \vdots & \vdots \end{bmatrix},\ B = \begin{bmatrix} \vdots \\ -\vec{p}_k \\ \vdots \end{bmatrix}$$

AX = B

Code:

```
X = numpy.linalg.lstsq(A,B)
```

Now we have a 6-d vector that consists of $\vec{p}_t\ and\ \vec{p}_{pivot}$

Validation and Results

Tuesday, October 26, 2021 7:31 PM

To verify the result, we have the following unit testing:

1) Verify that the registration algorithm calculates a reasonable R
2) Verify that the registration algorithm gives a correct frame by using debug files
3) Verify that the $\overrightarrow{C_{i(expected)}}$ we compute is correct by comparing with the $\vec{C}$ from debug-output file
4) Verify that the output of pivot calibration functions , $\vec{p}_t$, is the same as(or very close to) the actual values given in the debug output files.

1. To verify that the registration algorithm calculates a reasonable R by using the quaternion method we explained
   a. We use the data set name "pa1-debug-a" as our testing input
   b. By using the coordinates of $\overrightarrow{d_i}$ as the first point set input and coordinates of $\overrightarrow{D_i}$ in the first frame as the second point set input, the R that algorithm returned is a reasonable rotation matrix. To prove that, we calculae the det(R) and R$\boldsymbol{R^T}$

```
R =
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]


determinant of R = 1.0
R*R_transpose =
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

   As shown in the screenshot, R return a identity matrix and det(R) = 1 and R$\boldsymbol{R^T}$ = R$\boldsymbol{R^T}$ is an identity matrix

   We also test it by using the coordinates of $\overrightarrow{a_i}$ as the first point set input and coordinates of $\overrightarrow{A_i}$ in the first frame as the second point set input.

```
R =
[[ 1.    -0.012  0.003]
 [ 0.012  1.    -0.028]
 [-0.002  0.028  1.   ]]


determinant of R = 1.0
R*R_transpose =
[[ 1.000e+00 -8.132e-19  4.337e-19]
 [-8.132e-19  1.000e+00 -3.469e-18]
 [ 4.337e-19 -3.469e-18  1.000e+00]]
```

   As shown in the screenshot, R has det(R) = 1 and the result of R$\boldsymbol{R^T}$ = R$\boldsymbol{R^T}$ is an identity matrix

2. To verify that the registration algorithm gives a correct frame:

   We test it by using the coordinates of $\overrightarrow{a_i}$ as the first point set as input and coordinates of $\overrightarrow{A_i}$ in the first frame to test the output

   We test frame by calculate $\overrightarrow{\boldsymbol{A\ test_i}} = \boldsymbol{F_A} \cdot \overrightarrow{\boldsymbol{a_i}}$
   Then we compare the $\overrightarrow{A\ test_i}$ with $\overrightarrow{A_i}$ in the first frame given in the "pa1-debug-a-calreading.txt"

   $\overrightarrow{A_i}$ in the first frame given in the "pa1-debug-a-calreading.txt"

```
 209.06,   210.44, -1289.25
 209.70,   203.55, -1039.34
 205.98,   460.33, -1282.35
 206.62,   453.44, -1032.45
 459.04,   213.54, -1289.80
 459.68,   206.65, -1039.90
 455.96,   463.42, -1282.91
 456.60,   456.53, -1033.00
```

   $\overrightarrow{A\ test_i}$

```
209.06 210.44 -1289.25
209.7 203.55 -1039.34
205.98 460.33 -1282.35
206.62 453.44 -1032.45
459.04 213.54 -1289.8
459.68 206.65 -1039.9
455.96 463.42 -1282.91
456.6 456.53 -1033.0
```

Error = $\overrightarrow{A\ test_i}$ - $\overrightarrow{A_i}$ = 0

By computing the error between $\overrightarrow{A\ test_i}$ and $\overrightarrow{A_i}$, we get 0 error, which proves that the $\overrightarrow{A\ test_i}$ are the same value as $\overrightarrow{A_i}$, which verify that the registration algorithm gives a correct frame.

3. Results of computation of the expected values $\overrightarrow{C\_(i(expected))}$ for the $\overrightarrow{C}$

To further verify that the registration algorithm works, we used the debug files by reading the "calbody" and "calreading" txt file and then compare our result of C expected with the C value in the output txt file in each dataset. We calculate the average error of X, Y, Z separately in each data set. The comparison results are shown in the below table.

To calculate the average error of X, Y and Z in each dataset, we used the following equation:

Average error of X = $\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{abs}(\boldsymbol{Cexpected}[\boldsymbol{i}]_x - \boldsymbol{C}[\boldsymbol{i}]_x)$

Average error of Y = $\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{abs}(\boldsymbol{Cexpected}[\boldsymbol{i}]_y - \boldsymbol{C}[\boldsymbol{i}]_y)$

Average error of Z = $\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{abs}(\boldsymbol{Cexpected}[\boldsymbol{i}]_z - \boldsymbol{C}[\boldsymbol{i}]_z)$

Where N is the total number of C_expected value we get in each dataset

Table: Average error in each data set

Dataset	Average error of X in C_expected	Average error of Y in C_expected	Average error of Z in C_expected
pa1-debug-a	0.0021	0.0025	0.0027
pa1-debug-b	0.2546	0.2285	0.2496
pa1-debug-c	0.1453	0.2303	0.1372
pa1-debug-d	0.0050	0.0050	0.0055
pa1-debug-e	0.6561	0.6782	0.9503
pa1-debug-f	0.8640	0.7728	0.7704
pa1-debug-g	0.5993	0.7878	0.9564

Discussion of the results:

As we can see in the table, for Dataset A, when there is no EM distortion, EM noise or OT jiggle. The C_expected we calculated has very small error, which is less than 2 decimals, with the C values in the output txt file. These error can be neglected since the data in output file are only 2 decimals. These small error might be due to floating points. This proves that our 3D point set to 3D point set registration algorithm works correctly. In this ideal situation, the algorithm returns the result as we expected.

From dataset b to g, we can see that there are some errors between C_expected and the C value from the output. This also make sense, since from dataset b to g, there are either EM distortion, EM Noise or OT jiggle in the data that we used. Those noise will affect the registration algorithm we developed in this programming assignment. However, the error are generally small which is acceptable when there are noise in the dataset. We believe our result is corrected as we have proved that the registration algorithm and computation of C_expected are both correct in the previous verification.

The detailed output from debug dataset a-g are in the "OUTPUT" directory.

The detailed output for unknown data set h, i, j, k are shown in the tabular summary of the results and in the "OUTPUT"

directory.
Thus, in general, our algorithms works correctly.

4. Results of "Pivot" Calibration

Calibration for EM probe:

Dataset	$\vec{p}_{tip}$ Returned by Program	Expected $\vec{p}_{tip}$	Error
A	[205.88, 195.77, 193.11]	[205.88, 195.77, 193.11]	[0, 0, 0]
B	[205.07, 194.32, 201.72]	[205.07, 194.32, 201.72]	[0, 0, 0]
C	[200.36, 196.76, 207.22]	[200.36, 196.76, 207.22]	[0, 0, 0]
D	[191.09, 191.29, 208.32]	[191.09, 191.29, 208.32]	[0, 0, 0]
E	[205.42, 204.48, 194.78]	[205.42, 204.49, 194.78]	[2.12e-03, -7.90e-03, 8.36e-05]
F	[209.93, 201.4 ,200.62]	[[209.92, 201.4, 200.62]]	[0.01, 0, 0]
G	[196.04, 196.12, 199.07]	[196.03, 196.12, 199.07]	[0.01, 0, 0]

Calibration for Optical probe:

Dataset	$\vec{p}_{tip}$ Returned by Program	Expected $\vec{p}_{tip}$	Error
A	[404.19, 390.64, 192.65]	[404.19, 390.64, 192.65]	[0, 0, 0]
B	[409.97, 406.38, 195.08]	[409.97, 406.38, 195.08]	[0, 0, 0]
C	[402.96, 408.05], 198.29]	[402.96, 408.05], 198.29]	[-3.29e-05, 4.96e-03, -4.62e-03]
D	[390.48, 391.81, 205.75]	[390.48, 391.81, 205.75]	[0, 0, 0]
E	[396.37, 396.95, 197.21]	[396.37, 396.95, 197.21]	[0, 0, 0]
F	[399.25, 407.71, 206.18]	[399.25, 407.7, 206.18]	[0, 0.01, 0]
G	[403.72, 408.4, 193.51]	[403.72, 408.4, 193.51]	[0, 0, 0]

Discussion of the results:

Since there are distortion in dataset B-G, result of calibration is off by a tiny amount in some cases.
However, throughtout debug data set A-G, the error between calibrated EM/Optical probe tip position and the actual value is very acceptable $(\leq 0.01\ \text{mm})$ which proves program correctness for registration and calibration.

A tabular summary of the results obtained for unknown data

Wednesday, October 27, 2021 18:26

To show the result obtained from unknown data,

We calculate the average error in F_A in each data by using the following fomula:

Error of F_D in each frame = $\frac{\sum_{i=1}^{N_D}(\overrightarrow{D_i} - F_D \cdot \overrightarrow{d_i})}{N_D}$

Error of F_D in each dataset = $\frac{\sum_{i=1}^{N_D}(\text{Error of } F_D \text{ in each frame})}{N_{Frames}}$

Error of F_A in each frame = $\frac{\sum_{i=1}^{N_A}(\overrightarrow{A_i} - F_A \cdot \overrightarrow{a_i})}{N_A}$

Error of F_A in each dataset = $\frac{\sum_{i=1}^{N_A}(\text{Error of } F_A \text{ in each frame})}{N_{Frames}}$

Dataset	*Average Error in* F_A [x, y, z]	*Average Error in* F_D [x, y, z]
pa1-unknown-h	[0.0018, 0.0021, 0.0020]	[0.0016, 0.0026, 0.0019]
pa1-unknown-i	[0.0022, 0.0023, 0.0022]	[0.0020, 0.0023, 0.0018]
pa1-unknown-j	[0.0021, 0.0018, 0.0022]	[0, 0, 0]
pa1-unknown-k	[0.0019, 0.0022, 0.0018]	[0.0024, 0.0026, 0.0023]

Discussion of Result:

As shown in the table, the error of F_A and F_D are very small for these unknown dataset. These error are due to the unknown noise in these dataset. The error are small enough to be accept, as we proved out registration works correctly and algorithm can successfully compute right $\overrightarrow{C_{i(expected)}}$. Thus, we believe in these unknown dataset, our algorithm also compute the correct $\overrightarrow{C_{i(expected)}}$.

Calibration for EM probe:

Dataset	$\vec{p}_{tip}$ Returned by Program
H	[209.47,195.42, 216.93]
I	[206.3, 200.56, 194.23]
J	[191.07, 190.53, 210.24]
K	[191.1, 201.09, 187.52]

Calibration for OP probe:

Dataset	$\vec{p}_{tip}$ Returned by Program
H	[394.59, 399.97], 192.83]
I	[404.07, 398.23, 203.91]
J	[397.66, 408.18, 202.79]
K	[402.19, 403.11, 197.89]

Discussion of Result:
According to our validation result with Debug data sets, we expect the outcome of the program using unknown data sets to be very close to the real value if the scales of distortion in unknown data sets are similar to those in debug data sets.