Programming Asignment 2

Suryansh Shukla
sshukla8@jhu.edu
Zhiyuan Ding
zding20@jhu.edu

November 9, 2021

1 Contribution of each person

In the first design,Zhiyuan Ding works on Q1, Q2, write the main part of the report. Suryansh Shukla works Q3 to Q6.

But as Suryansh Shukla cannot work out the correct pivot calibration function, Zhiyuan Ding did **all of the works from Q1 Q6 with his own codes and write most part of the reports** except few words in 2.3.

2 Math approaches

2.1 Calibration of distortion

In this part, an distortion calibration method should be given based on the data pairs. In this program, we utilize the Bernstein polynomial to finish the fitting process.

The main part of this fitting process can be divided into 3 steps:

- Normalize the original data. This is done by first find out the range of corresponding data set. Namely, for data set x, we find the maximal value x_{max} and minimal value x_{min} in x. Then, we normalize the original data x by

$$x = \frac{x - x_{min}}{x_{max} - x_{min}}$$

  Thus the value of x is bounded between 0 and 1.

- Compute the Bernstein polynomial for the normalized data. In this problem, the data is 3D coordinates data (x, y, z) and let the maximal order of Bernstein function is N. Then in the Bernstein polynomial has $(N+1)^3$ entries, the value of ijh-th value in polynomial is:

$$F(x,y,z)_{i,j,h} = B_{N,i}(x)B_{N,j}(y)B_{N,h}(z)$$

  And the computation of $B_{N,i}(x)$ is:

$$B_{N,i}(x) = C_N^i(1-x)^{N-i}x^i$$

  After that, a vectorization of the $R^{(N+1)^3}$ tensor need to be done so that for M sample points, the obtained Bernstein matrix is in the size $R^{M\times(N+1)^3}$

- Then the polynomial coefficient matrix C is found by applying least square error method for the sample points. If the corresponding processed data is (X,Y,Z), then C in $R^{(N+1)^3\times 3}$ is fitted by solving:

$$\begin{bmatrix} & \cdots & \\ F_{000}(x_m,y_m,z_m) & \cdots & F_{NNN}(x_m,y_m,z_m) \\ & \cdots & \end{bmatrix} \begin{bmatrix} C^x_{000} & C^y_{000} & C^z_{000} \\ & \cdots & \\ C^x_{NNN} & C^y_{NNN} & C^z_{NNN} \end{bmatrix} = \begin{bmatrix} & \cdots & \\ X_m & Y_m & Z_m \\ & \cdots & \end{bmatrix}$$

The parameters C and the corresponding Bernstein function construct the distortion calibration function together.

2.2 Frame transformation relations in PA2

In this part, we briefly review the relationships we solved in PA2.

- In Q1, based on what we have done in PA1 Q3, we compute the frame transformation between EM tracker and optical tracker F_D and the frame transformation between calibration object and optical tracker F_A. Then we can get the expect values by:

$$C_{expect} = F_D^{-1} F_A c$$

- In Q2, with the obtained C_{expect} and measured C from EM tracker, we can compute a distortion correction function $f_B(C) = C_{expect}$. We use the Bernstein polynomial formation to fit this distortion correction function.

- In Q3, we first use f_B to get the corrected EM pivot data $G_{correctedempivot} = f_B(G_{empivot})$. Then we repeat the pivot calibration process in PA1 Q5 get the position of probe tip $\vec{p}_{tip}$.
- In Q4, we use f_B to get the corrected EM fiducial data $G_{correctedfiducial} = f_B(G_{fiducial})$. Then frame from Q3 is used to get the position of b_j relative to the EM tracker by: $B_j = F_G \vec{p_{tip}}$
- In Q5, just compute $b_j = F_{reg} B_j$
- In Q6, we first use f_B to get the corrected test data $G_{correctedem-nav} = f_B(G_{em-nav})$, then we get the tip location by $F_{reg} G_{correctedem-nav}$

2.3 Pivot Calibration

- Since the unknown variable are offset and (x_0, y_0, z_0)

$$\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} offset_x \\ offset_y \\ offset_z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} offset_x \\ offset_y \\ offset_z \\ 1 \end{bmatrix} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

$$r_{00} \cdot offset_x + r_{01} \cdot offset_y + r_{02} \cdot offset_z - 1 \cdot x_0 + 0 \cdot y + 0 \cdot z = -t_x$$

$$r_{10} \cdot offset_x + r_{11} \cdot offset_y + r_{12} \cdot offset_z + 0x_0 - 1 \cdot y + 0 \cdot z = -t_y$$

$$r_{20} \cdot offset_x + r_{21} \cdot offset_y + r_{22} \cdot offset_z + 0x_0 + 0 \cdot y - 1 \cdot z = -t_y$$

This equation can be written as

$$M \cdot \begin{bmatrix} offset_x \\ offset_y \\ offset_z \\ x_0 \\ y_0 \\ z_0 \end{bmatrix} = N$$

For many samples

$$M = \begin{bmatrix} r_{00}^1 & r_{01}^1 & r_{02}^1 & -1 & 0 & 0 \\ r_{10}^1 & r_{11}^1 & r_{12}^1 & 0 & -1 & 0 \\ r_{21}^1 & r_{22}^1 & r_{23}^1 & 0 & 0 & -1 \\ r_{00}^2 & r_{01}^2 & r_{02}^2 & -1 & 0 & 0 \\ r_{10}^2 & r_{11}^2 & r_{12}^2 & 0 & -1 & 0 \\ r_{21}^2 & r_{22}^2 & r_{23}^2 & 0 & 0 & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

N can be written as

$$N = \begin{bmatrix} -t_x^1 \\ -t_y^1 \\ -t_z^1 \\ -t_x^2 \\ -t_y^2 \\ -t_z^2 \\ \vdots \end{bmatrix}$$

$$M \cdot \begin{bmatrix} offset_x \\ offset_y \\ offset_z \\ x_0 \\ y_0 \\ z_0 \end{bmatrix} = N$$

We can solve the above equation using least square estimation.

3 Algorithm steps

3.1 Q1

As some of wrong understanding of the definition of frames in the process of finishing PA1, here we rewrite the function codes for Q1 based on the distortion calibration function from PA1.

For each frame k, the frame transformation based on data pairs (a_k, A_k) and (d_k, D_k) is computed and then we obtain the frame F_{C_k}. With F_{C_k} we get the value $C_{expect}[k]$. The algorithm is shown in 1.

Also in our design, we consider to process multiple data sets once, so an outer layer iteration is added. Output for each data set is stored in specified place.

Algorithm 1 Designed framework for C_expect output function

```
Input: The path of original data including $A, a, D, d, c$, path to save output data,
1: for $frame = 1$ to $N_{frame}$ do
2:     Data loading $A, a, D, d$
3:     Compute the frame transformation $F_C[k]$
4:     Compute $C_{expect}[k] = F_C[k]C[k]$
5: end for
Output: $C[1], C[2], ..., C[k]$ into the corresponding save path.
```

3.2 Q2

The fitting process of Bernstein polynomials is just as the above three-step process in math approaches. We consider to build one distortion correction function based on samples from all frames in each data set, i.e. in our debugging 125×27 samples in each correction process.

For each coordinates the maximal and minimal values are also preserved to normalized the following EM measured data.Algorithms is shown in Algorithm 2.

Algorithm 2 Designed framework for distortion correction functions

```
Input: The path of original data $C$, path of $C_{expect}$, designed order of Bernstein function $N_B$
1: Normalize each coordinates
2: Compute the Bernstein function in size $R^{M \times (N_B+1)^3}$
3: Fitting the Bernstein coefficient by least square manner
Output: The corresponding Bernstein coefficient, the bounding range for each coordinates
```

3.3 Q3

In Q3 we first need to get the distortion free empivot data using what the distortion correction function we obtained. Then based on what we've done in PA1 Q3 and Q5, we can get the registration relationship between g and G and then use the pivot calibration function to obtain the position of p_{tip}. An algorithm is shown in 3

```
Algorithm 3 Designed framework for frame pivot calibration
Input: the original EMPIVOT data $C$,
  1: Use the distortion correction function to process G
  2: Get corresponding frames for each $(G_k, g)$ pairs
  3: Use the pivot calibration function to find out the position of tip $p_{tip}$
Output: position of $p_{tip}$, $g$
```

3.4 Q4

Similar to Q3, we just need to get the distortion corrected data $G_{fiducial}$ first and then get the frames F_{G_K}. Then we use the relationship:

$$B_j = F_{G_K} \vec{p_{tip}}$$

to get the corresponding coordinates B_j

3.5 Q5

Points registration based on PA1 Q2.

3.6 Q6

This is also a synthetic application for previous functions: first we get the distortion corrected G_{nav} based on distortion correction functions; then we get the frame based on PA1 Q5; finally a frame transformation based on F_{reg} is utilized to get the coordinates in CT system.

4 Overview of the program

In figure 1 is the overview of the program. The program is divided into three parts: The PA2-DEMO.py contains the direct utilization to solve from Q1 to Q6. With a direct changes in the input path and output path in current working directory and a given name of data set, the user can run the DEMO file to work out all of the 6 questions.

And the running of PA2-DEMO.py depends on the functions in PA2-functions.py . Some of the main functions in PA2-functions.py are:

- C-expect-output function to obtain the result C_{expect} in each dataset. This is only used in Q1.
- distortion-correction-Bernstein function to compute the distortion correction function based on the Bernstein polynomial fitting process. This function is only used in Q2.
- distortion-correction-output function is used to generate the corrected data for each data set based on the computed Bernstein function. This function is used in Q3, Q4 and Q6.
- frame-pivot-calibration function works on the pivot calibration for every dataset to solve Q3. Please notice that distortion-correction-output function for $G_{empivot}$ should be operated in advance to get the corrected data.
- fiducial-coordinate-em function works on getting the coordinates for B based on the registration process in PA1 Q5 and a simple frame transformation.Please notice that distortion-correction-output function for $G_{fiducial}$ should be operated in advance to get the corrected data.
- F-reg function is used to get the frame F_{reg}.

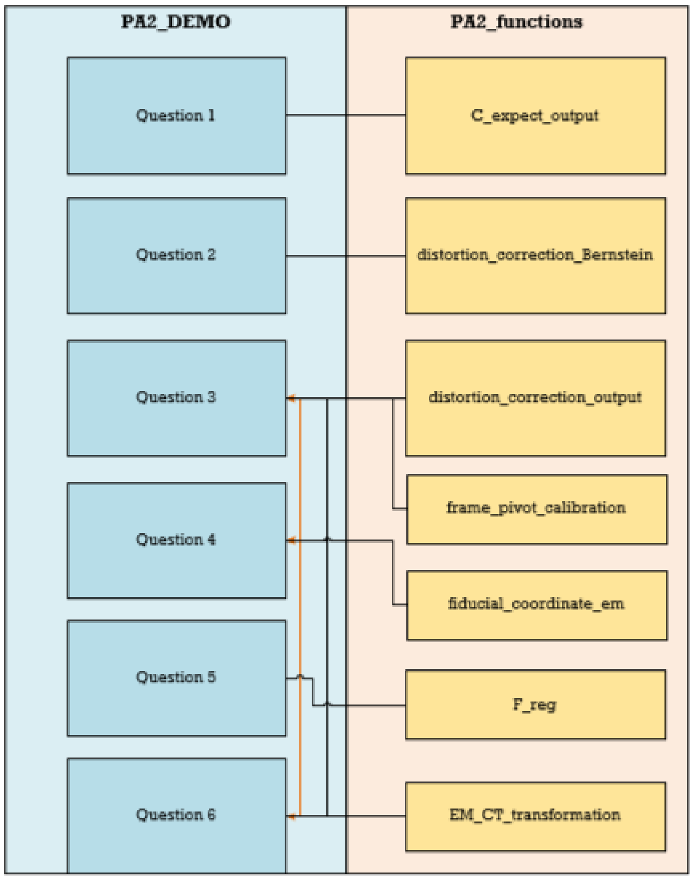


Figure 1: Overview of the PROGRAMS

- EM-CT-transformation function is based on the a registration process in PA1 Q5 and two frame transformation.Please notice that distortion-correction-output function for G_{em-nav} should be operated in advance to get the corrected data.

There are also some other functions used in the middle processed part in PA2-functions.py, such as scale-to-box function to normalize the original data between $[0, 1]$, the Bernstein-function to calculate the Bernstein entries and the bounding-box-with-known to normalize the input data based on pre-defined bounds etc.

Some of the functions in PA1-functions.py are also used to get compute the middle result, such as the pivot-calibration function in PA1 Q5 is used in Q3, Q4 and Q6 and the points-registration function is used in Q5 to obtain the frame F_{reg}.

5 Verification

5.1 Q1 and Q2

First here is a check for the fitting result of distortion correction functions. In order to measure the fitting ability, we use the corresponding C as the input and then compare the differences between the output C'_{expect} and original C_{expect}. The bias for each debugging and unknown data set is shown in Table 1. We can see that the fitting bias is quite negligible.

Table 1: Bias of the fitting Bernstein polynomial

data set	x	y	z
a	8.06638073e-06	8.16635375e-06	8.15228038e-06
b	0.07644098	0.07843445	0.07556178
c	0.0003221	0.00020386	0.00034414
d	5.13621787e-05	3.77952942e-05	5.63526000e-05
e	0.0044521	0.00479623	0.00428147
f	0.01930686	0.02838082	0.01876695
g	0.00151365	0.00135032	0.00089366
h	0.01344407	0.01376392	0.01533265
i	0.01306351	0.01302681	0.01298529
j	0.0146722	0.0146059	0.01393672

Table 2: Output Tabular for unknown data set

Name	Description
NAME-C-expect.npy	computed C_{expect} value for each data set
	125×27 points for 125 frames
NAME-coordinate-range.npy	Corresponding bounding range for each data set.
	3×2 matrix contains min max for (x, y, z)
NAME-function-coeff.npy	C parameters in obtained Bernstein polynomials.
	This is a $(B_n + 1)^3 \times 3$ matrix, where B_n is the order of polynomials.
NAME-G-empivot.npy	obtained result for EMPIVOT data
	after the distortion correction process
NAME-G-em-fiducials.npy	obtained result for EM fiducial data after
	the distortion correction process
NAME-G-em-nav.npy	obtained result for EM test data after
	the distortion correction process
NAME-B-em.npy	the obtained position of B_j
NAME-F-reg.npy	the computed frame between B and b
NAME-G-ct-nav.npy	final output for the points in CT coordinates
NAME-F-reg.npy	computed frame between B and b
NAME-g-em.npy	corresponding g from the first frame relative to the probe
NAME-p-pivot.npy	result p_{pivot} and p_{tip}

6 Q3 to Q6

In the middle part we do not have a known standard result for each variables from Q3 to Q5 and the variables from Q3 to Q6 are used one after each other, so a comparison between the obtained tip location in CT image from Q6 and the known standard output from pa2-debug-NAME-output2.txt can show the correctness of our program.

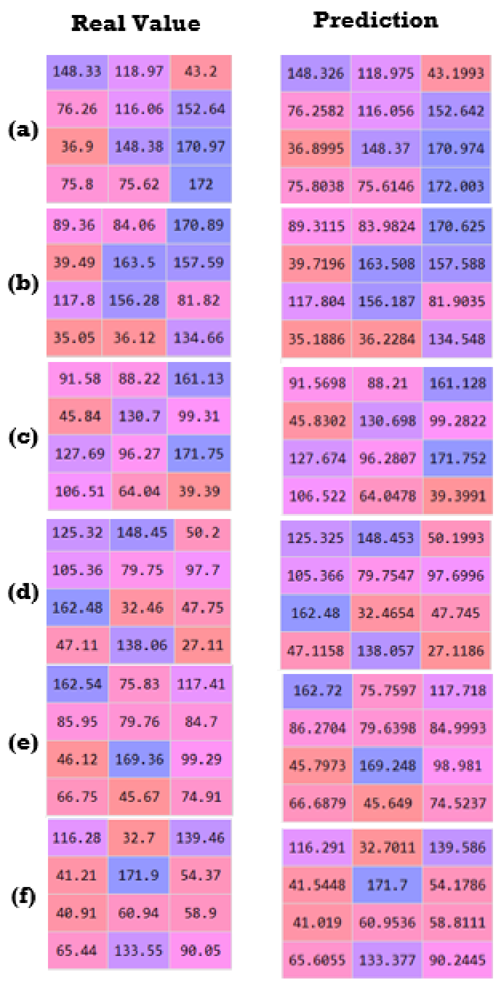
In this thought, we directly compare this two parts as shown in Figure 2, the actual maximal bias between the actual CT value and the registered value is less than 0.5 percentage so we can show the effectiveness of our method.

7 Result from unknown data

The output files for unknown data is shown in Table 2

References

1. Numpy package https://numpy.org/



	Real Value			Prediction		
(a)	148.33	118.97	43.2	148.326	118.975	43.1993
	76.26	116.06	152.64	76.2582	116.056	152.642
	36.9	148.38	170.97	36.8995	148.37	170.974
	75.8	75.62	172	75.8038	75.6146	172.003
(b)	89.36	84.06	170.89	89.3115	83.9824	170.625
	39.49	163.5	157.59	39.7196	163.508	157.588
	117.8	156.28	81.82	117.804	156.187	81.9035
	35.05	36.12	134.66	35.1886	36.2284	134.548
(c)	91.58	88.22	161.13	91.5698	88.21	161.128
	45.84	130.7	99.31	45.8302	130.698	99.2822
	127.69	96.27	171.75	127.674	96.2807	171.752
	106.51	64.04	39.39	106.522	64.0478	39.3991
(d)	125.32	148.45	50.2	125.325	148.453	50.1993
	105.36	79.75	97.7	105.366	79.7547	97.6996
	162.48	32.46	47.75	162.48	32.4654	47.745
	47.11	138.06	27.11	47.1158	138.057	27.1186
(e)	162.54	75.83	117.41	162.72	75.7597	117.718
	85.95	79.76	84.7	86.2704	79.6398	84.9993
	46.12	169.36	99.29	45.7973	169.248	98.981
	66.75	45.67	74.91	66.6879	45.649	74.5237
(f)	116.28	32.7	139.46	116.291	32.7011	139.586
	41.21	171.9	54.37	41.5448	171.7	54.1786
	40.91	60.94	58.9	41.019	60.9536	58.8111
	65.44	133.55	90.05	65.6055	133.377	90.2445

Figure 2: Result of Q6 output compared with real value in CT

2. Bernstein polynomials https://en.wikipedia.org/wiki/Bernstein polynomial