Programming Assignments 1 & 2 601.455 and 601/655 Fall 2021
Please also indicate which section(s) you are in (one of each is OK)

Instructions and Score Sheet (hand in with answers)

Name Haoyu Shi	Name Jiahe Xu
Email hshi25@jh.edu	Email jxu109@jh.edu
Other contact information (optional)	Other contact information (optional)
I have followed the rules in completing this assignment (signature) Haoyu Shi	I have followed the rules in completing this assignment (signature) Jiahe Xu

Grade Factor		
Program (40)		
Design and overall program structure	20	
Reusability and modularity	10	
Clarity of documentation and programming	10	
Results (20)		
Correctness and completeness	20	
Report (40)		
Description of formulation and algorithmic approach	15	
Overview of program	10	
Discussion of validation approach	5	
Discussion of results	10	
TOTAL	100	

CIS1 PA2 Report

By Jiahe Xu and Haoyu Shi

1. Overview:
    - Note that all files are meant to be used with pa2 files, it will only be able to read files with "pa2-" prefixes!
    - Added files:
        - Distortion_correction.py
        - Improved_em_tracking.py
        - Fiducials.py
        - Clean.sh.py
        - Eval.ipynb
    - Adjusted files:
        - Cartesian.py
        - Registration_3d.py
        - Eval_all.py
        - Eval.py
        - Main.py
        - Run.sh
        - Clean.sh
2. Structure of coded files:
    a. Files required for running and evaluating:
        i. Run.sh, clean.sh
        ii. Main.py, eval.py
    b. General functions used in almost all:
        i. Cartesian.py
        ii. Pivot_calibration.py
        iii. Registration_3d.py
    c. For specific steps:
        i. Distortion_correction.py: Step 1 and 2 in guidelines

        Dependencies:

- Registration_3d.py
- Cartesian.py

ii. Improved_em_tracking.py: Step 3 in guidelines

Dependencies:

- Pivot_calibration.py
- Cartesian.py
- Registration.py
- Distortion_correction.py

iii. Fiducials.py: Step 4 and 5 in guideline

Dependencies:

- Pivot_calibration.py
- Cartesian.py
- Registration.py
- Distortion_correction.py
- Improved_em_tracking.py

3. Required Running environment:

   Same as before, mostly basic packages and numeric libraries such as numpy.

4. Algorithm approach:

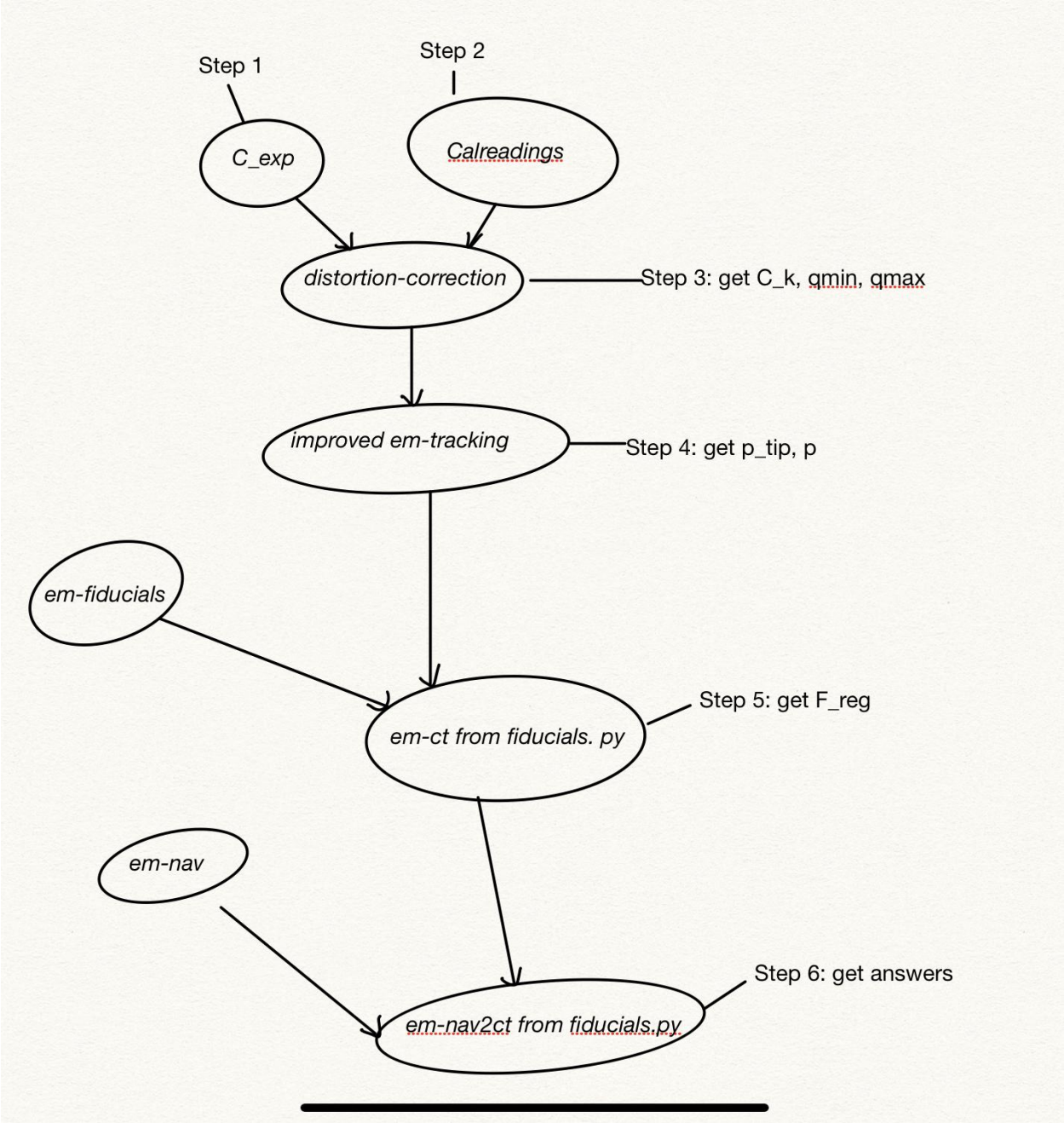
   We did the program mostly following the assignment guidelines:

   a. Get C_exp numbers, the ground truth. We decided to use the numbers given in output1 due to the fact that our C_exp during the evaluation of the previous homework, turned out to still have an amount of error.
   b. Read the calreadings file. This step reads the sensor data and prepares it for calculating correction.
   c. Correction. During this step, we devised a file distortion_correction which is based on the Bernstein Polynomial given in class. The ground truth p here is the C_exp values while the sensor data q is the processed values from calreadings. The return value c_k is then used to calibrate the empivot data and the same process for em_tracking is performed again.

At the end of this step, we will be able to get p_tip: The position of the tip of the probe relative to the tool, p_dimple: The em position of the tip, and p which is the data points in the coordinate system defined when calculating p_tip.

d. Calculating F_reg. This step corresponds to the step 4 and 5 in the given assignment guidelines. We first calculated the p_tip of the probe using improved_em_tracking. The values p_tip and p will be useful in the following steps. We then read the required files containing data from fiducials(em_fiducials). These points are in the em tracking system. Therefore, we first calculate their actual point position by calculating the frame transformations between the corresponding frame and the previous points in the system of pivot_calibration.(value p returned by pivot_calibration). The transformation of each frame is then used to transform the points to their corresponding locations in the em tracking system by multiplying it with p_tip. The outcome will be the fiducials' locations in the em tracking system. With the positions in the em system and the positions in the ct system(obtained by reading the files ct-fiducials), we can use our previously defined function registration_3d to get F_reg
e. Calculating test points. This step corresponds to the last step which calculates data read from em_nav files. The data is first transformed from frames into actual points in the em_system. It is then undistorted to get accurate values. The final step is to use the previously defined function: point_transform and the calculated F_reg to get the final position in the ct image.

The following figure explains the workflow described above:



In code, C_k is represented as “X” in all files except “distortion_correction.py” for convenience.

5. In depth explanation of files:
    a. Added files:
        - Distortion_correction:

This mainly contains functions calculating the Bernstein Polynomials required for calculating the distortion correction matrix. Our method is based on the interpolation review note of the class. Functions inside:

- get_bern_comps(N,k,v):
  Calculates an individual part of the equation based on the formula given in the interpolation review lecture, namely using $B_{N,k}(v) = \binom{N}{k}(1-v)^{N-k}v^{k}$.
- scale(X, X_min, X_max):
  Rescale a set of points(X), according to limits: X_min and X_max

$$ScaleToBox(x, x^{min}, x^{max}) = \frac{x - x^{min}}{x^{max} - x^{min}}$$

- Berntensor_matrix(points , qmin, qmax, order):
  This function is used to build the Bernstein polynomials. We need to rescale the points and then compute the coefficient matrix with get_Bern_comp().

$$F_{ijk}(u_x, u_y, u_z) = B_{5,i}(u_x)B_{5,j}(u_y)B_{5,k}(u_z)$$

- distortion_correction(q,p,order)
  “q” is the set of points that are distorted, “p” is the set of points that are accurate, we hope to turn “q” to “p” with coefficient “X”.
  This function aims to compute the weight corresponding to elements of Bernstein polynomials. We follow the method mentioned in class, turning the problem to “AX=B” form,so we can use the least square method to compute “X”. “A” is the Bernstein polynomial coefficient matrix.
  By the way, **q corresponds to u** in the following formulation.

$$\begin{bmatrix} & \vdots & \\ F_{000}(\vec{u}_s) & \cdots & F_{555}(\vec{u}_s) \\ & \vdots & \end{bmatrix} \begin{bmatrix} c^x_{000} & c^y_{000} & c^z_{000} \\ \vdots & \vdots & \vdots \\ c^x_{555} & c^y_{555} & c^z_{555} \end{bmatrix} \approx \begin{bmatrix} & \vdots & \\ p^x_s & p^y_s & p^z_s \\ & \vdots & \end{bmatrix}$$

- data_correction(data,X,Nframe,Ng,qmin,qmax)
  This function is used to correct distorted points(data), with coefficients X we get from distorttion_correction(), Nframes and Ng are used to illustrate the structure of points set (data). We first compute the corresponding Bernstein polynomial coefficient matrix B_mat, and then use it with coefficient X, we can get the corrected points.
  Parameter “data” corresponds to “q” in the following formulation.
  We make Bij into a matrix “B_mat”.

$$\vec{\mathbf{p}} = CorrectDistortion(\vec{\mathbf{q}})$$
$$\{ \quad \vec{\mathbf{u}} = ScaleToBox(\vec{\mathbf{q}}, \vec{\mathbf{q}}^{\min}, \vec{\mathbf{q}}^{\max})$$
$$\text{return} \sum_{i=0}^{5}\sum_{j=0}^{5}\sum_{k=0}^{5} \vec{\mathbf{c}}_{i,j,k} B_{5,i}(u_x) B_{5,j}(u_y) B_{5,k}(u_z)$$
$$\}$$

- Improved_em_tracking: This file utilizes the distortion_correction function mentioned above. It first uses the **C_exp** files calculated from cal_readings and calbody to calibrate the **em_pivot** data, then carry out the same procedures as the previous em_tracking file. This is also the function which covers problem 3.
  We will save the result of **P_tip P_dimple** and
  p(markers’ positions in initial frame)
  for other functions, and we save them as a list called “empivot”

- fiducials: This file contains two functions which covers the problems 45 and 6 respectively.
    - em_ct: The function used for calculating F_reg. It takes the frames and sets up a coordinate system at the mid point of the given data points. The improved_em_tracking.py file also now returns p, the point set which was created in pivot_calibration. This allows us to have a consistent coordinate system for the **P_tip** value. Then, using **P_tip** and the p values, we are able to generate the positions of the given fiducial points in the em_tracking system. Using the register_3d function written before, we can input the point sets in the em and ct systems. The output will be the calculated **F_reg** value.

      To get fiducial position in EM frame, we first use distortion_correction() to correct all the points we get from EM. Then, we need to compute the transformation **F_i** from current fool frame to initial tool positions ***p*** that we get from improved_em_tracking.

      F_i = registration(*p* , corrected_G_positions)

      So fiducial_i = F_i * P_tip

      The ctfiducials_data are points of fiducials in CT frame, they are correct.

      The **pivot_se**t are points of **fiducial_i**

      ctfiducials_data = F_reg * pivot_set

      We can find F_reg by:

      F_reg = registration(pivot_set , ctfiducials_data)
    - em_nav2ct: The function used for actual calculation of the test points.

      First, we use distortion_correction() to correct all the points we get from EM.Then,we need to compute the

transformation **F_i** from current fool frame to initial tool positions ***p*** that we get from improved_em_tracking.
F_i = registration(*p* , corrected_G_positions)
So fiducial_i = F_i * P_tip
The **pivot_se**t are points of **fiducial_i**
pivot_ct = F_reg * pivot_set

- clean.sh: Added script for easier cleanup of generated output and eval-log files.
- Eval.ipynb: Used for graphing the evaluation results.
- main.py
  In main.py to avoid computing the weights of Bernstein polynomial coefficient repeatedly, we save the result of distortion_calibration(), and pass “X,qmin,qmax” to functions that need to use them, same things to the result of improved_em_tracking(): “empivot”, and the result of em_ct(): “F_reg”

b. Adjusted files:
   - Cartesian: The file for most of the general math and point transformation package was adjusted.
     - combinations: Added a function which gets the result of N choose k. This is used for get_bern_comps functions
   - Registration_3d: This file was adjusted in an attempt to further reduce the error.
     - error_correction: It takes the F calculated before as F0 and looping until the delta value is small enough

6. Discussion of evaluation results

We implemented results for 3 of the steps.
All evaluations below assess from four aspects of the differences: average, variance, max and min. The main focus is on average as the others do not give specific information on the results comprehensively. Especially min which has no

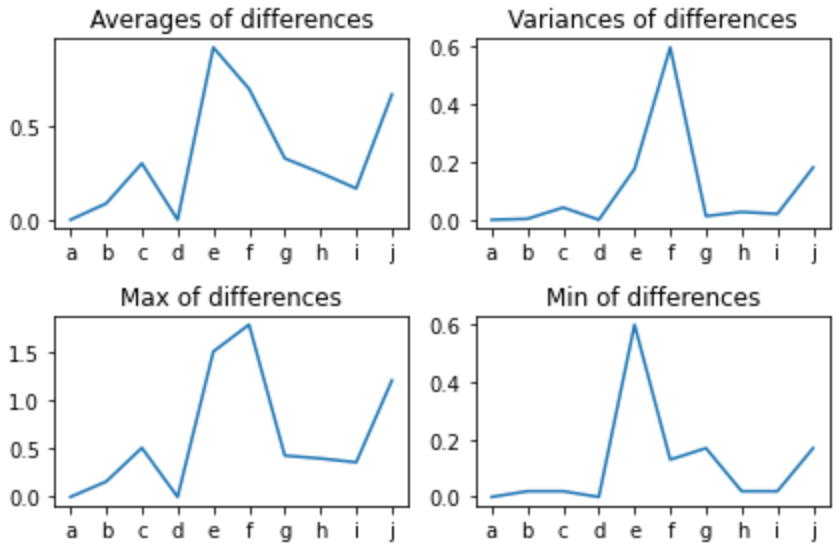
meaning in F_reg as the last row is always 0 0 0 1 which makes the graph a flat line of 0. The data given had noise which was indicated in the graph in the handouts. They are divided in three aspects: EM distortion, EM Noise and OT jiggle.

Data set name	EM distortion	EM Noise	OT jiggle
pa1-debug-a	0	0	0
pa1-debug-b	0	X	0
pa1-debug-c	X	0	0
pa1-debug-d	0	0	X
pa1-debug-e	X	0	X
pa1-debug-f	X	X	X
pa1-debug-g	X	X	X
pa1-unknown-h	X	X	X
pa1-unknown-i	X	X	X
pa2-debug-a	0	0	0
pa2-debug-b	0	X	0
pa2-debug-c	X	0	0
pa2-debug-d	0	0	X
pa2-debug-e	X	X	X
pa2-debug-f	X	X	X
pa2-unknown-g	X	X	X
pa2-unknown-h	X	X	X

a. The improved em_pivot:

We tested and compared our data with the improved_em_tracking function.

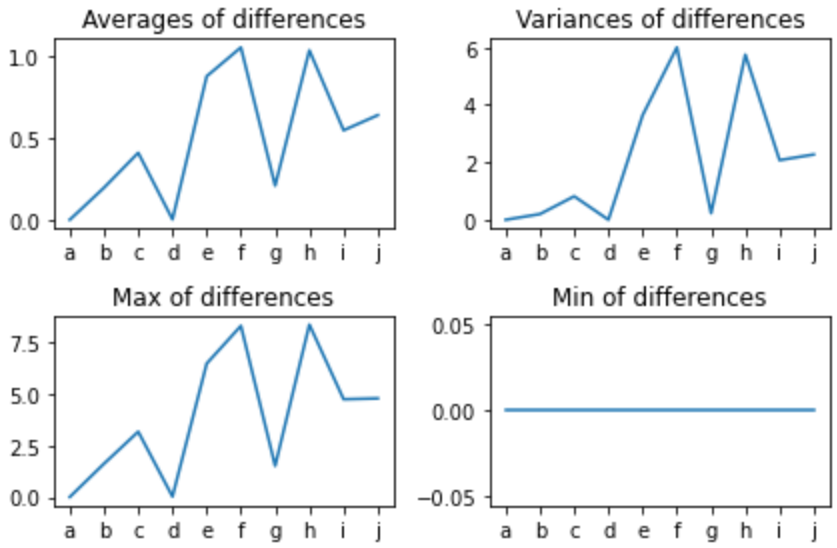
This is a set of plots describing the data of em_pivots after applying the new improved_em_tracking function. It can be seen that although the points e and f have max difference values above 1 which is not exactly as expected, the others are mostly in an accepted range. It shows that the values a and d have little to no noise, and produces results that are almost equal to the ground truth. This result shows better accuracy compared to the previous homework results. The previous homework(pa1) produced results of averages over 1 under similar circumstances including noises in all 3 aspects. The other datasets despite the fact that they have noise in all three aspects as well showed good results.



b. F_reg:

As mentioned above, min cannot be used for evaluation as the result is always 0 due to the fact that the last row will always be 0 0 0 1 and produces this seemingly "perfect" result for min. In addition, F_reg was not directly given in a data file like the other 2, it was calculated by doing register_3d between the points transformed from frames in em_nav files and output2. The error in our registration_3d might also be a contributing factor.

Despite both b and c having either noise or distortion in the data, the results produced are slightly different. C seems to have a small increase among a,b ,c and d, the four files that have the least noise and distortion. The remaining files produced relatively higher errors except for g. G has been the one data file which has consistently better performance than the other files with all three aspects of noise.



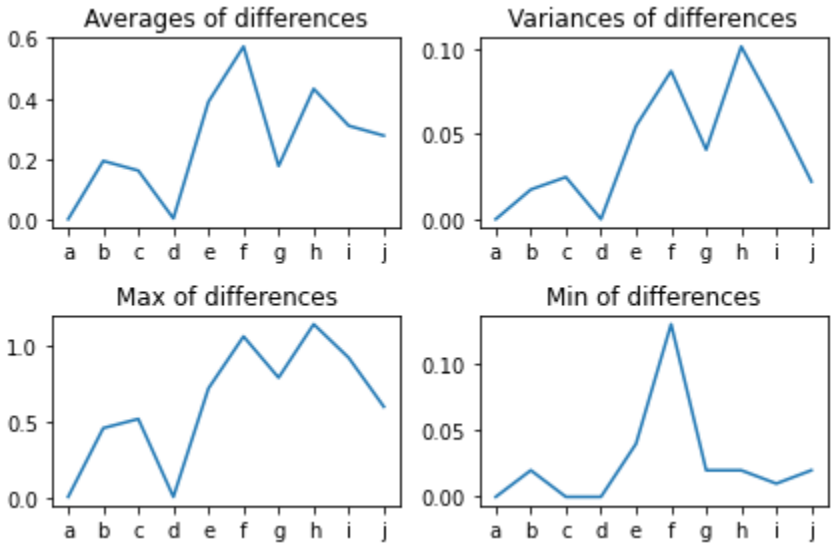
c. The results of the ct_points are plotted below. Due to the fact that pa2-debug-a dataset lacked the file containing data for opt-pivot, its opt_pivot evaluation was removed and the file used to evaluate it was specifically adjusted.(main_a.py)

The results show the same pattern among all the evaluated fields of the differences including averages, variances, max values and min values. It shows that a and d were the data set which performed the best. This can probably be explained with the graph provided in the homework alongside the data. The dataset d has neither em distortion nor em noise while a has zero noise in any aspect. Its data avoided getting errors accumulated and propagated along when using the em tracking system.

The datasets b and c are relatively low but noise still exists which caused slightly larger errors compared to d. This is possibly because of the existence of only 1 noise in the data. Among the other two, c has a lower average of differences. A possible explanation would be that the noise present in c is em_distortion instead of noise. The distortion was fixed in step 3 and all following steps using the distortion_correction function. While the noise can only be mitigated through the least squares solutions calculations.

The remaining datasets: e, f, h, i are very close except for g and j which have a relative drop of differences(rise in accuracy) but still not comparable to a, b, c and d. This is directly linked to the presence of noise in all three aspects.

Data set name	EM distortion	EM Noise	OT jiggle
pa1-debug-a	0	0	0
pa1-debug-b	0	X	0
pa1-debug-c	X	0	0
pa1-debug-d	0	0	X
pa1-debug-e	X	0	X
pa1-debug-f	X	X	X
pa1-debug-g	X	X	X
pa1-unknown-h	X	X	X
pa1-unknown-i	X	X	X
pa2-debug-a	0	0	0
pa2-debug-b	0	X	0
pa2-debug-c	X	0	0
pa2-debug-d	0	0	X
pa2-debug-e	X	X	X
pa2-debug-f	X	X	X
pa2-unknown-g	X	X	X
pa2-unknown-h	X	X	X



From all above results, there are some patterns that can be noted:

- Files without noise are always better performing
- Files with less noise in em areas can potentially influence positively the results
- G has been consistently better among the files with all three aspects of noise. This might be due to the fact that the noise in g is relatively small.

Participation:

We both developed and debugged all the functions and code in this assignment.