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	

1. Overview of program

- **Files used in final build:**
    - cartesian.py
    - pivot_calibration.py
    - registration_3d.py
    - distort_calibration
    - em_tracking.py
    - optical_tracking.py
- **Files used during development process:**
    - 4.ipynb
    - 5.ipynb
    - 6.ipynb
- **Folders:**
    - DATA
    - OUTPUT
    - eval_logs
- **Files used for generating output:**
    - run.sh

2. Required Running environment:

- Inside of the submission, there is a env.yml file that can be used to create the environment we used for developing and running our code.

- The packages involved include:
    - genericpath: Used to check whether the required files exist.
    - numpy: The common numerical library for creating arrays and doing matrix operations.
    - glob: Writing the required path used in reading and writing files.
    - pathlib: For getting the path of objects

    - argparse: For parsing the input string given to the main.py file.
    - csv: For writing the output file
    - os: Used to deal with path and various directory traversal actions.
    - copy: Duplicating necessary arrays safely

3. In depth explanation of files

- **Initial data used for calculations:**
  - **DATA** folder:
    The files inside of this folder were provided to us along with the homework. This is the main directory we set our program to read from.

- **Files used in final build of the program**
  - **main.py:**

    Main program is used to combine different programs and achieve the designated objectives. Given various inputs in the command line including path to the data, preferred output folder, selected files, it outputs the results and evaluations of our data compared to the given answers.

  - **eval_all.py and eval.py:**

    Both functions are just used for evaluating results.

  - **cartesian.py:**

    Created first and includes multiple functions:

    1. isRot: This function determines whether the given R matrix is a rotation matrix or not.
    2. b. Ri: This function calculates the inverse of a Rotation matrix.
    3. c. get_R and get_t: Returns the rotation and translation of a frame.
    4. concat_frame: Given a rotation and a translation, return a frame.
    5. Fi: Returns the inverse of a frame.
    6. f. points_transform: Given a point and a transformation, return the resulting points.

- **registration_3d.py:**

  Inputs:

  start: the set of points we taken as start points

  goal: the set of points we taken as goal points

  return a 4*4 transform matrix F, where $F \cdot start = goal$

  The way we compute F is according to the formula on rigid3d3dcalculations.pdf p9, "*Direct Techniques to solve for R*".

  When det(R) is -1, we use the original least square method to solve the problem. Since the original least square method is in the form of AX=B find the best X. We need best F that $F \cdot start = goal$, so we need to reform A, and use turn F into X which is a 9*1 unknown vector. After we get the vector, we can change the shape in to a 3*3 rotation matrix

- **distort_calibration.py:**

  After we read in the data, we get d,a,c point set and D point set, A point set and C points set in each frame. And we can compute FD and FA transformations with registration_3d(), as described in the assignment file. And use FD_inv transformation to get C_expeted in each frame.

- **em_tracking.py:**

  For each frame, we use the method mentioned in class, and given as following:

  1.Use d(data d) in calbody.txt to get Fd for each set of points in each frame.

  2. Use the Fd to update H points in each frame.

  3. Apply pivot_calibration to comput P_dimple.

- **optical_tracking.py:**

  As said in the questions and guidelines in the homework handout, this is similar to the problem of em tracking.

1. We obtained d_i from the calbody files and D_i from the optical files(namely Nd).
2. Fd can in turn be obtained by applying the function registration_3d() to the two values given above. We did the calculation on all D but found that the Fd obtained are all the same. Thus, we used the first set of Fd.
3. Transform the given Nh coordinates using Fd.
4. Apply a method similar to the em_tracker above to the processed data.

- **pivot_calibration.py:**
  This code is used to calculate the vector P_dimple. The required inputs are different sets of points : data, Np, number of points in each frame and Number of frames: Nframe.
  With these data, we applied the method mentioned in class: using the first "frame" of pivot calibration data to define a local "probe" coordinate system and followed by the method below
  1. Compute the midpoint:P0 of the observed points.
  2. Translate the observations relative to this midpoint, get different transformations, and collect all the frames
  3. Seperate the rotation and translation components
  4. Apply the function solve_lsq() defined in the same file to get the final answer. The detail of formulating A and B matrices in the least square problem is described in **pivot_calibration.py.**

4. Discussion of validation methods and results

All the methods we use in our program are mentioned in class, as we have detailed explanation on the last section, and here is the result:
In debug data:
Average in error of C_expect: **9.55767396e-01**
Average in error of em_pivot: **3.89696970e-01**
Average in error of optical_pivot: **9.09090909e-04**

These are the results I was able to obtain from running the eval_all.py file. It shows that the most accurate results are of the optical tracking data. Although the other two still look acceptable, it is not quite the same when looking at them individually.
In the folder eval_logs, for example, the evaluations for the file "unknown-h" shows an average error of em_pivot to be close to 1 and its C_exp even reached an average error of over 5. This is alarming but no issues in our coding was found. In addition, most of the other files show an average of C_exp and em_pivot well below 1. Most of the opt_pivot data are even 0 or small enough to be considered 0(e-07).
The results show that our program is able to make the correct calculations under most circumstances. The results of the file "unknown-h" could be considered as an exception or an outlier.

5. Deliverables:

- **OUTPUT folder:**
  We created this folder using the functions in main.py. The files contained inside are in the same format as the output files given to us.

- **Eval_logs folder:**
  This folder contains all evaluation files created by us for each of the 11 files from "a-debug" to "k-unknown". It shows the difference, variance and max/min between each of the original and created files.

6. Reusability and modularity

Due to the fact that Programming assignment 2 has strong correlation with the first and that not much time is left for it, we designed the structure and models so that they can be reused. Most of the functions, except for the output functions used to evaluate and store results of this particular assignment, are in its own python file which can be reused on other assignments that have a large amount of calculations that are similar. Each standalone package or python file serves as a part of common calculations instead of programs specifically for this assignment.